TUOTANTOLINJAN TOIMINNAN OPTIMOINTI KONEOPPIMISEN AVULLA

LAHDEN AMMATTIKORKEAKOULU
Tekniikan ala
Kone- ja tuotantotekniikan koulutusohjelma
Tuotantopainotteinen mekatroniikka
Opinnäytetyö
Kevät 2011
Jesse Luhti

Lahden ammattikorkeakoulu
Kone- ja tuotantotekniikan koulutusohjelma

LUHTI, JESSE

Tuotantolinjan toiminnan optimointi koneoppimisen avulla

Tuotantopainotteisen mekatroniikan opinnäytetyö, 43 sivua, 13 liitesivua

Kevät 2011

TIIVISTELMÄ

Tämä opinnäytetyö käsittelee viilujen jatkamislinjan optimointia koneoppimisen keinoin. Työn toimeksiantajana on Koskisen OY. Työn suoritus tapahtui vuosien 2009, 2010 ja 2011 aikana.

Opinnäytetyön tarkoituksena on parantaa viilujen jatkamislinjan kapasiteettia poistamalla linjan loppuosan jakolaitteiden toiminnan aiheuttama pullonkaula. Käytettyjä koneoppimisen keinoja ovat evoluutio-oppimismenetelmät, kuten geneetinen algoritmi ja geneettinen ohjelma. Tuotantolinjasta luodaan simulointimalli, jossa linjan loppuosan viilujen jakolaitteille kokeillaan erilaisia toimintamalleja, joista geneettisellä algoritmillä saadut tulokset siirretään tuotantolinjan logiikkaan. Geneettisellä ohjelmalla etsitään yksittäiselle jakolaitteelle sopivaa toimintalogiikkaa, mutta hyvää menetelmää ei löydy ja geneettinen ohjelma vaatii lisätutkimuksia.

Geneettisen algoritmin tuloksena saatu vakiojako toimintamalli poistaa kapasiteetin pullonkaulan lähes täysin. Vakiojakoa täydennetään automaattisella syöttötahdin laskennalla, joka vähentää operaattoreilta vaadittavaa säätötyötä sekä parantaa linjan kapasiteettia. Täyden kapasiteetin saavuttamiseksi vaadittaisiin keskitettyä päätöksentekoa, jossa linjaa ohjaavalle logiikalle luotaisiin ylemmän tason järjestelmä.

Avainsanat: optimointi, koneoppiminen

Lahti University of Applied Sciences
Degree Programme in Mechanical and Production Engineering

LUHTI, JESSE

Optimizing production line with artificial intelligence

Bachelor's Thesis in mechatronics 43 pages, 13 appendices

Spring 2011

ABSTRACT

This study optimizes the scarf-jointing of veneer using artificial intelligence (AI). The study was commissioned by Koskisen OY and it was made between 2009 and 2011.

The purpose of this study was to improve the capacity of a veneer scarf-jointing production line by removing bottleneck caused by actuators which divide veneers to joint presses. The AI methods used comprise genetic algorithm and genetic program. A simulation model was created from an real world production line and different approaches were tested for optimal efficiency. The genetic algorithm produces good results with constant veneer distribution order and required modifications were made to the program of the production line. However the genetic program has failed to produce a suitable solution and requires more research.

The genetic algorithm solution removes capacity bottle neck almost completely. This solution was improved with automatic veneer infeed pace calculation which both helps operators work and improves the capacity by feeding veneers at precisely right pace. There is however some room for improvement which would require a system that has information of whole production line's operation, and could make correct choice every time.

Key words: optimization, artificial intelligence

SISÄLLYS

1 JOHDANTO

Opinnäytetyössä etsitään koneoppimisen keinoin ratkaisuja Koskisen OY:n jatkoslinjan 2 kapasiteettiongelmaan. Opinnäytetyön painopiste on linjan suurimmassa yksittäisessä pullonkaulassa, joka on viilujen jakaminen puristimille. Tavoitteena on saada linjalle käyttöön jako-ohjelma, joka jakaa viiluja tasaisesti puristimille eikä aiheuta kapasiteetin pullonkaulaa.

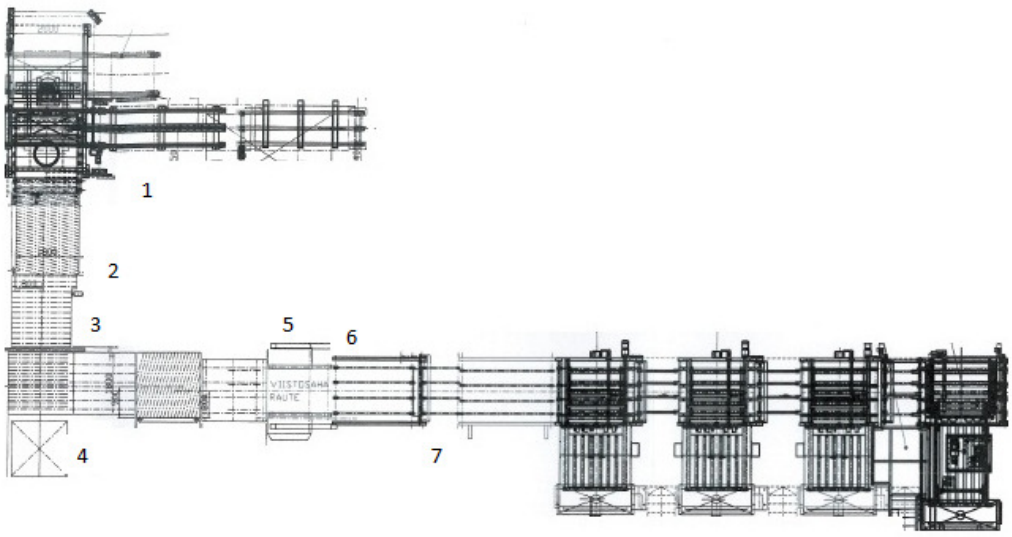
Vanerilevyn keskellä käytetään kahdenlaisia viiluja: jatkettuja ja saumattuja. Normaalisti jatkettuja viiluja menee vain hieman enemmän kuin saumattuja, mutta tiettyihin tuotteisiin tarvitaan huomattavasti enemmän jatkettuja. Kun jatkettuja viilua tarvitaan pidemmän aikaa enemmän kuin mihin vaneritehdas on suunniteltu, muodostuu jatkamisesta pullonkaula levyjen valmistukselle. Viilujen jatkamisen kapasiteetin lisäämiseksi tehtiin vuoden 2008 alkuosalla investointi Koskisen OY:n jatkoslinja 2:lle, jossa ohjausjärjestelmä uusittiin ja linjaan lisättiin yksi jatkospuristin. Opinnäytetyön ajankohta sijoittuu investoinnin vastaanoton jälkeiseen aikaan. Opinnäytetyössä pyritään korjaamaan linjalla esiintyviä vikoja, jotka aiheuttavat myös kapasiteetin alenemista. Linjan logiikkaohjelmassa oli jakolaitteiden osalta vakavia toiminnallisia ongelmia, jotka estivät täyden kapasiteetin saamisen. Neljä puristinta pystyi käsittelemään suuremman määrän viiluja kuin mitä jakolaitteet pystyivät niille syöttämään.

Koskisen on vuonna 1931 perustettu suomalainen, kansainvälisesti toimiva perheyritys. Koskisen valmistaa ja markkinoi mekaanisen metsäteollisuuden tuotteita rakennus-, rakennuspuusepän-, huonekalu- ja kuljetusvälineteollisuudelle. Yli 70 vuoden aikana yritys on kehittynyt kansainväliseksi puun ammattilaiseksi. Koskisen liiketoimintayksiköt ovat puunhankinta, sahateollisuus, levyteollisuus ja taloteollisuus. Levyteollisuuteen kuuluvat Järvelän toimipisteen vaneri- ja lastulevyteollisuus sekä Hirvensalmella sijaitseva koivutuoteteollisuus. Vuonna 2010 Koskisen OY:n liikevaihto oli 181 milj.€, josta viennin osuus 55 %. Henkilöstöä oli vuonna 2010 yhteensä 994. (Koskisen OY 2011.)

2 VIILUJEN JATKAMISLINJA

Jatkamisen tarkoituksena on valmistaa viiluja, joiden pituusmitta on suurempi kuin sorvipöllien mitta tekemällä vinoliitos viilun päähän. Näin voidaan liimata suurikokoisia levyjä. (Koponen 2002, 127.)

Viilut syötetään linjalle pinoista syy-suunta edellä. Linjalle syöttämisen jälkeen viilut tasataan ja reunasaha suoristaa viilun reunat. Kamerajärjestelmä tunnistaa reiät ja halkeamat. Mikäli viilun laatu ei ole riittävä, ohjataan viilu haaske-kuormaan, josta viilu menee seuraavaan työvaiheeseen alenevassa laadussa. Laadultaan riittävät viilut ohjautuvat viistesahoille, joissa viilun päät viistetään liitosta varten. Toiseen viisteeseen levitetään liima ja viisteiden kunto tarkistetaan. Viilut, joiden viiste ei ole riittävän hyvä, ajetaan roskakuormaan, josta ne menevät hakkuriin. Hyväviisteiset viilut ohjataan kohti linjan loppuosaa, jossa viilut jaetaan puristimille. Tämä opinnäytetyö keskittyy ainoastaan linjan loppuosaan, jonka muodostaa kolme jakolaitetta sekä hihnakuljettimet ennen puristimia. Puristimilla kelkka ottaa viilun risteysasemalta, kohdistaa kahden viilun viisteet kohdalleen ja vie ne puristimeen. Puristinpalkit ovat lämmitettyjä eli kyseessä on kuumapuristus. Puristuksen jälkeen viiluarkit mitataan ja leikataan haluttuun pituuteen ladontaa varten.



KUVIO 1. Jatkamislinjan layout

Kuviossa 1 merkityt kohdat:

1. syöttölaite
2. reunasaha
3. konenäköjärjestelmä
4. haaskelava
5. viistesahat
6. liimoitin
7. roskalava.

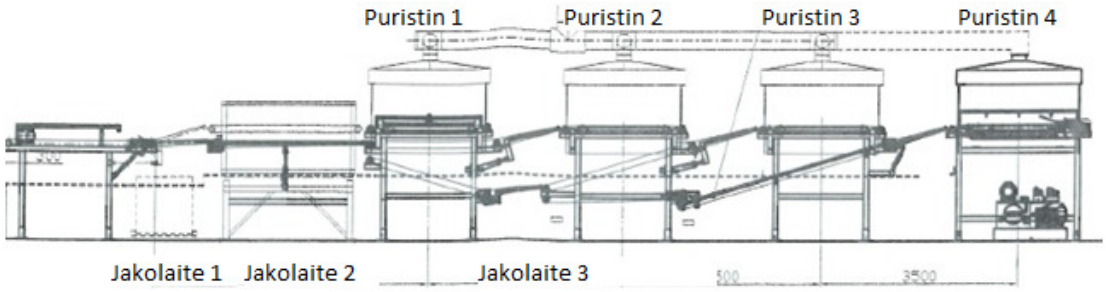
2.1 Haaskeen ja roskien vaikutus kapasiteettiin

Jatkamislinjan kuljettimista suurin osa on suoralla sähkömoottrikäytöllä eli niiden nopeutta ei voida muuttaa ajon aikana. Tämä tarkoittaa sitä, että kun viilu hylätään pois linjalta joko haaske- tai roskakuormaan, jää siihen kohtaan tyhjä aukko, jota ei voida ajaa kiinni. Syöttötahdin ollessa vakio menetetään kapasiteettia, sillä yhtään hylättyä viilua ei voida korvata. Mikäli syöttötahti on säädetty hieman liian tiheäksi, esimerkiksi keskimääräisen hylkäysprosentin mukaan, käy toisinaan niin, että linjalle syötetään enemmän viiluja kuin mitä sinne mahtuu. Hylkäysprosentti vaihtelee suuresti syöttökuormittain, 2 ja 20%:n välillä, kuten tiedonkeruutiedoista voidaan todeta (LIITE 1). Mikäli linjalle syötetään liikaa viiluja, alkaa linjan alkupää pysähdellä ja odottaa, että loppupää ehtii kuluttamaan viilut. Jokaisesta pysähdyksestä seuraa liiman levityksen epäonnistuminen liimauksessa olevalle viilulle, koska levitys toimii vain tasaisesti liikkuvalle viilulle. Liiman levityksen epäonnistuessa viilu pitää ajaa roskiin ja viilua ei voida enää hyötykäyttää tehtaalla. Tästä syystä linjan operaattorit mieluummin säätävät syöttötahdin hieman liian hitaaksi ja takaavat näin katkottoman ajon, mutta menettävät kapasiteettia kuin aiheuttavat viilujätettä madaltaen linjan hyötysuhdetta.

2.2 Hyvän jako-ohjelman ominaisuuksia

Viilut jaetaan puristimille kolmella paineilmatoimisella jakolaitteella, jotka ovat käytännössä toisesta päästä nivelöityjä hihnakuljettimia. Jakolaitteet ovat erikokoisia ja –massaisia, sekä niitä liikuttavat sylinterit ovat erilaisia. Jokaisella jakolaitteella on siis oma, yksilöllinen liikeaikansa. Jako-ohjelman tulisi pystyä oh-

jaamaan nykyistä mekaniikkaa riittävän ajoissa, jotta jakolaite saa aikaa liikkua asemaan ennen kuin viilu saapuu kuljettimen päähän. Näin varmistutaan siitä, ettei jako itsessään muodostu kapasiteetin rajoittajaksi. Toinen vaatimus jako-ohjelmalle on jaon tasaisuus. Vuoron mittaisessa ajanjaksossa pientä eriarvoisuutta saa tapahtua, koska eri puristimilla on eri määrä häiriöitä ja kuorman vaihtoja, mutta kun kaikki puristimet ovat ajossa, tulisi viilujen jakautua kaikille puristimille tasaisesti. Tämä ominaisuus korostuu etenkin silloin kun syöttötahti säädetään alhaiseksi, jolloin kaikille puristimille ei riitä viiluja koko ajan. Näin voidaan joutua tekemään joskus raaka-aineesta tai linjan alkupäästä johtuvien ongelmien takia. Jakolaitteiden ja puristimien sijainti on esitetty kuviossa 2.



KUVIO 2. Jakolaitteiden ja puristimien sijainti jatkamislinjalla

2.3 Viilujen jaon merkitys linjan toiminnalle

Parhaimmillaan viilujen jako jatkoslinjalla on täysin näkymätön linjan käyttäjälle. Jakolaitteet ja niiden ohjaus eivät rajoita linjan kapasiteettia, ja jako-ohjelma pystyy nopeasti toipumaan lyhytkestoisista häiriötilanteista. Logiikkapäivityksen jälkeen linjalla ollut jako-ohjelma rajoitti linjan kapasiteettia merkittävästi sekä aiheutti ruuhka- ja muita häiriötilanteita linjalla.

Alkuperäinen jako-ohjelma perustui hetkittäiseen päätöksentekoon sillä ohjelman kierrolla, jolla viilu saapui jakolaitteen anturille. Ensimmäisen jakolaitteen osalta tämä aiheutti tilanteen, jossa jakolaite pyrki aina jakamaan puristimelle 1, jos siellä oli tilaa. Ylösjako oli ohjelman kierrossa ennen alasjakoa. Linjan alkuosan häiriöiden takia voi tulla tilanne, jolloin viiluja tulee noin 15 sekunnin välein linjan loppuosaan. Alkuperäinen jako-ohjelma jakoi nämä kaikki viilut puristimelle 1,

koska puristimen tahtiaika oli pienempi kuin viilujen väli. Sama ongelma toistui muillakin jakolaitteilla, mutta jakolaitteen 1 osalta ongelmat olivat kaikista merkittävimmät. Tähän ongelmaan etsittiin nopeaa ja helppoa korjausta tekemällä ohjelmaan lisäehto: jos kaikki neljä puristinta ovat ajossa, niin vain joka kolmannen viilun jako sallitaan puristimelle 1. Tällaisista rajoitteista aiheutuu muita ongelmia, jotka näkyvät häiriötilanteissa.

3 KONEOPPIMINEN

Koneoppimisen tarkoituksena on saada kone ratkaisemaan ongelma ilman, että ongelmaa täytyy koneelle täysin määritellä. Matemaattisen ratkaisun tapauksessa koko ongelma ja sen ratkaisu on hyvin tiedossa. Matemaattisista yhtälöistä voi nähdä miten eri tekijät vaikuttavat tulokseen. Monimutkaisemmissa ongelmissa, jossa matemaattinen ratkaisu on hyvin työläs tai mahdoton, voidaan yhdeksi ratkaisun etsimisvaihtoehdoksi ottaa koneoppiminen.

Seuraavissa luvuissa esitellään joitakin erilaisia menetetelmiä, joita harkittiin viilujen jako-ongelman ratkaisemiseksi. Neuroverkot ja päätöksentekopuut vaativat, että haluttu toiminta tiedetään ja voidaan opettaa koneelle. Geneettinen algoritmi vaatii, että ongelma voidaan muuttaa sellaiseen muotoon, jossa kromosomien koodaus kelpaa ratkaisuksi. Geneettinen ohjelma puolestaan pystyy etsimään erilaisia ratkaisuvaihtoehtoja, joiden rakenne ei ole alussa selvillä.

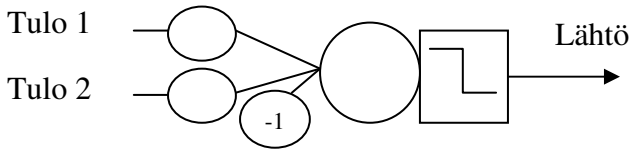
3.1 Neuroverkot

Neuroverkot ovat yksi koneoppimisen osa-alue. Neuroverkkojen toiminta perustuu neuroverkon kykyyn oppia datasta ja pyrkimykseen tehdä yleistyksiä. Neuroverkkoja voidaan käyttää esimerkiksi prosessien säätimien säätöhorisontin pidentämiseen arvioimalla tehtävien säätöjen vaikutusta prosessiin. Toinen mielenkiintoinen käyttökohde on luoda neuroverkolla virtuaalinen anturi. Tämä virtuaalinen anturi voi korvata fyysisen anturin paikoissa, joissa suureen suora mittaaminen on hankalaa.

Neuroverkon käyttö yleisesti jakautuu seuraaviin vaiheisiin: verkon rakenteen valinta, verkon alustus satunnaisilla arvoilla, opetusdatan syöttö, tuloksen laskenta ja virheiden korjaus. Opetusdataa syötetään niin kauan, kunnes oppiminen on saavuttanut halutun tason eli virhe muodostuu riittävän pieneksi tai todetaan, että verkko ei opi syötetystä datasta haluttua asiaa. Opetusvaiheen verkon toiminta tarkistetaan sellaisella datalla, joka ei ollut mukana opetuksessa. (Marsland 2009, 55.)

3.1.1 Perseptroni

Perseptroni on neuroverkon peruskomponentti, joka rakentuu muutamasta osasta. Perseptronin rakenne on esitetty kuvassa 3. Perseptronilla on n kappaletta tuloja, joilla jokaisella on oma painoarvonsa. Tulojen määrä riippuu sisään syötettävästä datasta eli jokaiselle datan ulottuvuudelle tarvitaan oman sisääntulonsa. Kaikki painotetut tulot ovat käytössä aktivointifunktiossa. Aktivointifunktion arvon perusteella päätetään, aktivoituuko perseptronin lähtö vai ei. (Marsland 2009, 19)



KUVIO 3. Perseptronin rakenne

Jokaisella perseptronilla on ns. Bias-tulo, jonka arvo on aina -1. Oppimisalgoritmi, joka oppimisen aikana muuttaa perseptronin tulojen painoarvoja, muuttaa myös Bias-tulon painoarvoa. Bias-tuloa tarvitaan, kun kaikki tulot ovat arvossa 0, mutta perseptronin lähdön halutaan aktivoituvan. (Marsland 2009, 22.)

3.1.2 Yksikerros perseptroni –verkko

Yksikerros perseptroni -verkko koostuu rinnankytketyistä tuloista ja rinnankytketyistä perseptroneista. Yleensä kaikki tulot on yhdistetty kaikkiin lähtöihin. Tulojen määrä valitaan sisään tulevan datan ulottuvuuksien mukaan, ja se voi olla sama tai eri kuin lähtöjen (perseptronien) määrä. Opetusvaiheessa lähtöjen tilaa verrataan oikeaan vastaukseen, ja mikäli lähdön tila on väärin, tulee perseptronien arvoja korjata niin, että tulos lähenee haluttua. (Marsland 2009, 20.)

Yksikerros perseptroni -verkko soveltuu esimerkiksi lineaarisiin luokitteluihin, joista esimerkkinä loogiset operaatiot, kuten JA ja TAI. Luokittelu perustuu kahdessa ulottuvuudessa suoraan, kolmessa ulottuvuudessa tasoon ja korkeammissa ulottuvuuksissa hypertasoon. (Marsland 2009, 32.)

3.1.3 Monikerros perseptroni –verkko

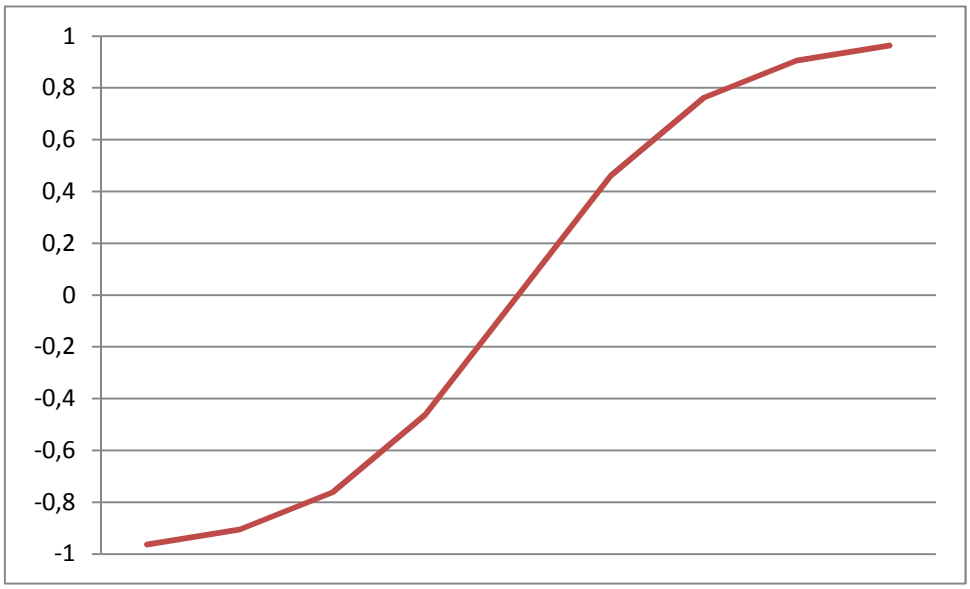
Yksikerroksiset verkot pystyvät siis lineaariseen luokitteluun ja monimutkaisempiin ongelmiin tarvitaan joko lisää ulottuvuuksia tai monimutkaisempia verkkoja. Monikerros perseptroni -verkoissa on yksi tai useampi piilotettuja kerroksia tulo- ja lähtökerroksen välissä. Jokaisessa kerroksessa on rinnakkain perseptroneja haluttu määrä. Nimitys piilotettukerros tulee siitä tosiasiasta, että näiden kerrosten arvoja ei ole mahdollista tarkastella suoraan. (Marsland 2009, 48.)

Jokaisen kerroksen perseptroni on yleensä yhdistetty kaikkiin seuraavan kerroksen perseptroneihin. Tämän lisäksi jokaisella perseptronilla on ns. Bias-tulo, joka jo aiemmin mainittiinkin. Monikerros perseptroni -verkon ulostuloa tarkasteltaessa opetusvaiheessa on helppo todeta, mitkä perseptronit eivät toimi oikein, kuten yksikerros verkossakin, mutta nyt korjattavien painoarvojen löytäminen on hieman monimutkaisempaa. (Marsland 2009, 48.)

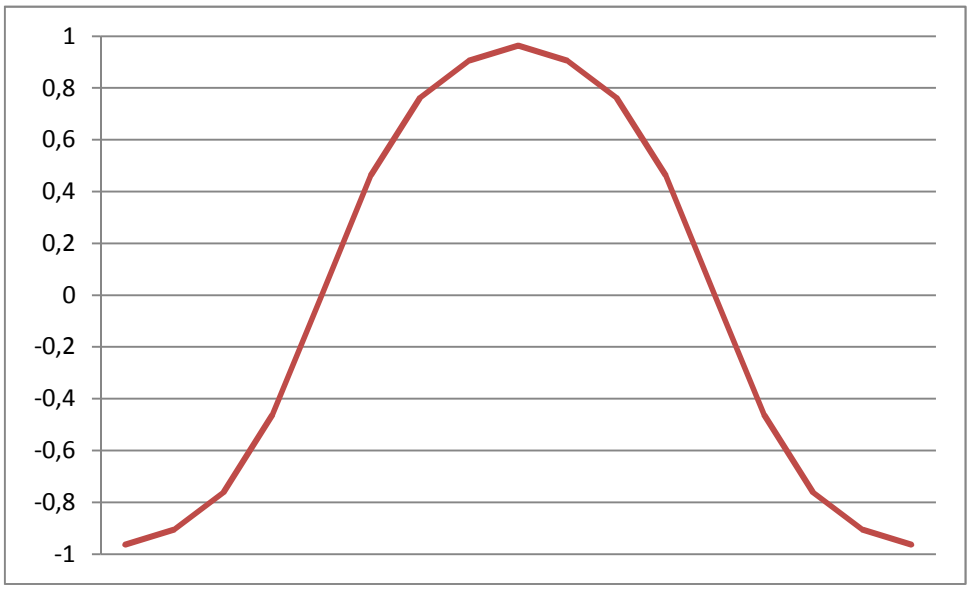
Kerrosten määrästä ja aktivointifunktiosta riippuen verkko kykenee erilaisiin ratkaisuihin. Yleisesti käytetty aktivointifunktio on hyperbolinen tangetti, joka on ns. sigmoid-funktio, mutta saturoituu välillä ± 1.

$$g(h) = \tanh(h) = \frac{e^{h} - e^{-h}}{e^{h} + e^{-h}}$$

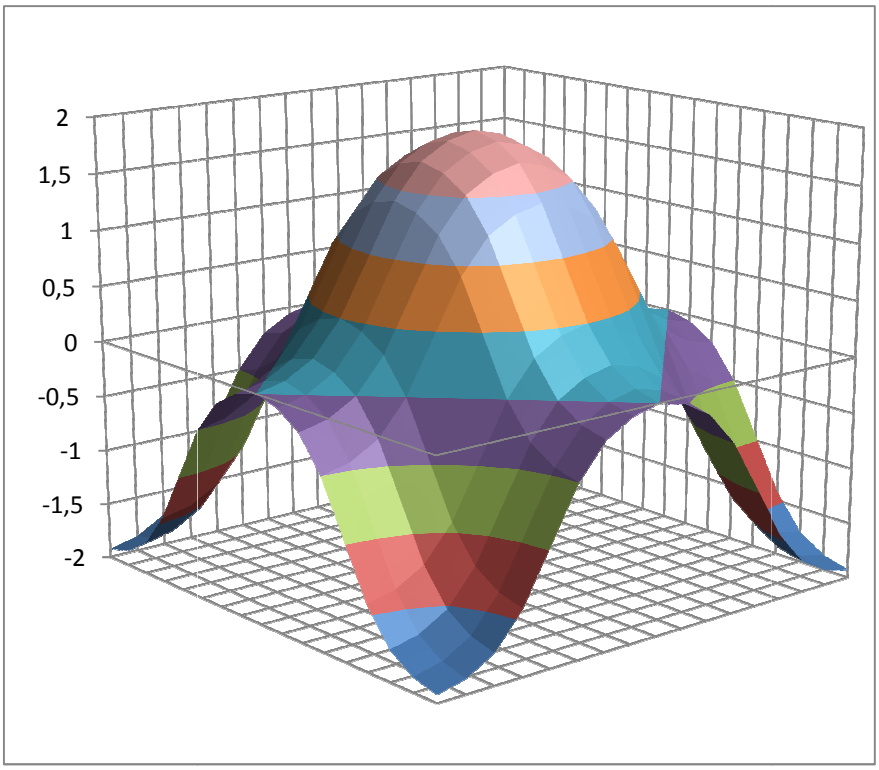
Kuvioissa 4,5 ja 6 on esitetty yhden, kahden ja kolmen peräkkäisen sigmoid-funktion mahdollistamat luokittimet. Yhdellä sigmoid-funktiolla voidaan sisään syötettävä data jakaa kahteen luokkaan, kahdella sigmoid-funktiolla kolmeen luokkaan ja kolmella sigmoid-funktiolla useisiin luokkiin tarpeen mukaan.



KUVIO 4. Yhden sigmoid-funktion mahdollistama luokitin



KUVIO 5. Kahden sigmoid-funktion mahdollistama luokitin



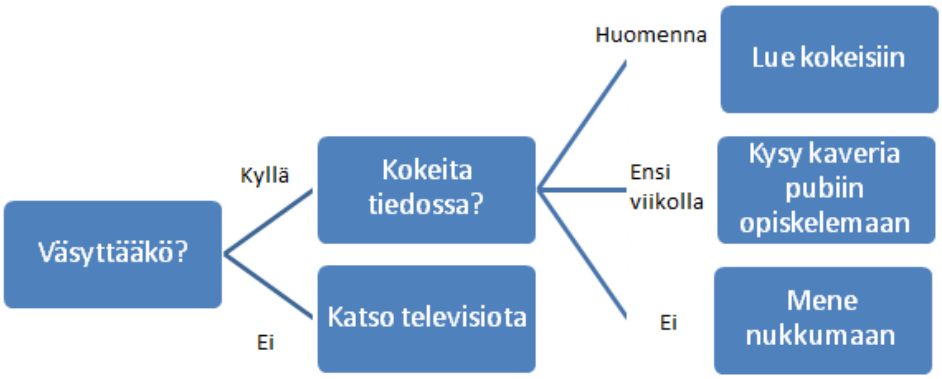
KUVIO 6. Kolmen sigmoid-funktion mahdollistama luokitin

3.2 Päätöksentekopuut

Päätöksentekopuiden luominen vie suhteellisen vähän laskentatehoa, mutta päätöksentekopuun käyttämisen kustannus on vielä pienempi. Tämä yksi päätöksentekopuiden vahvuus, koska tuloksia kysellään useammin kuin puuta luodaan ja monesti tulokset halutaan käyttöön heti. Toinen etu, joka päätöksentekopuilla on, että ne ovat hyvin helposti ihmisen ymmärrettävissä. Ihmiset luottavat tulokseen enemmän kun se ei tule ”mustasta laatikosta” vaan jostain, jonka he voivat käsittää. (Marsland 2009, 133.)

Päätöksentekopuussa lähdetään ylhäältä tai vasemmalta ja liikutaan kohti alareunaa tai oikeaa reunaa, siis ratkaisua, vastailemalla monivalintakysymyksiin. On tärkeää, että eniten informaatiota sisältävä kysymys sijaitsee ensimmäisenä ja kysymyksien merkitys vähenee kohti ratkaisua liikuttaessa. Yksi tapa määritellä ominaisuuden tärkeys on ajatella, kuinka paljon uutta informaatiota saadaan tietämällä ominaisuuden arvo. Ominaisuus, jonka arvon tietämällä saa eniten uutta informaatiota on tärkein ja tulisi siksi valita ensimmäiseksi kysymykseksi. Kuviossa 7 on esimerkki päätöksentekopuusta.

Päätöksentekopuut toimivat hyvin diskreeteillä luvuilla, mutta jatkuvilla luvuilla täytyy käytännössä tyytyä jonkinlaiseen luokitteluun. Luvut voidaan luokitella haluamallaan tarkkuudella, esimerkiksi suurempi tai pienempi kuin nolla, tai luokittelutarkkuus voi olla käytössä oleva laskentatarkkuus. Erilaisia raja-arvo käyttämällä päätöksentekopuut sopivat hyvin myös luokitteluun koneoppimisen lisäksi. (Marsland 2009, 133)

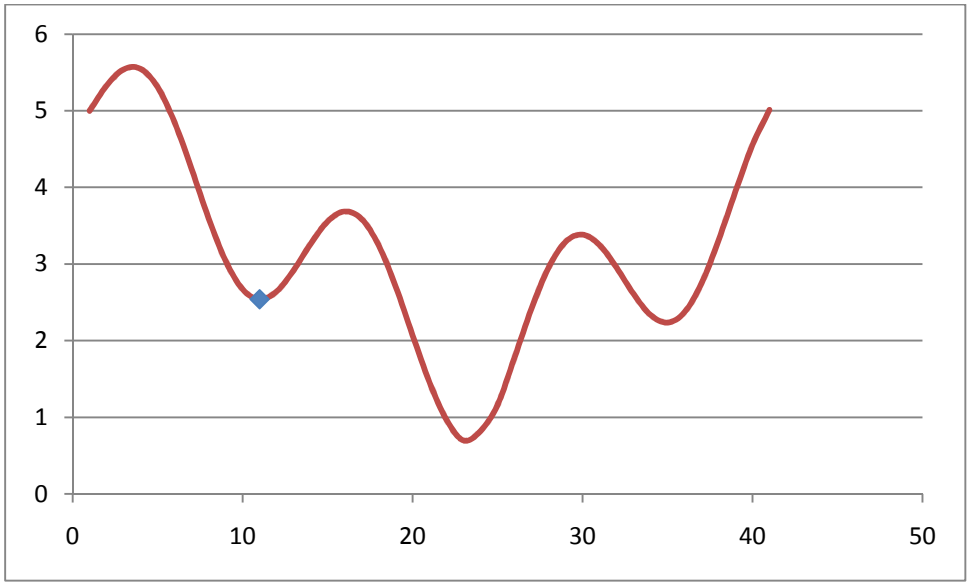


KUVIO 7. Esimerkki päätöksentekopuusta, joka kuvaa opiskelijan iltaohjelman suunnittelua

3.3 Evoluutio-oppimisen perusteet

Geneettisessä ohjelmoinnissa luodaan aluksi satunnainen populaatio erilaisia ratkaisuja, joita geneettinen ohjelma (engl. Genetic Program) stokastisesti muokkaa uusiksi ratkaisuvaihtoehdoiksi. Ajatuksena on, että hyviä ratkaisuja yhdistämällä saadaan luotua vieläkin parempia ratkaisuja. GP on, kuten luonnossakin, sattumanvarainen prosessi, eikä sen voi taata saavuttavan aina samoja tuloksia. Koeajoissa tarvitaankin useita toistoja parhaiden ratkaisujen löytämiseksi. Geneettinen ohjelma ei myöskään todennäköisesti löydä optimaalista ratkaisua vaan löytää hyvän, joka voi olla lähellä optimaalista. Geneettisellä ohjelmalla ratkaistavat ongelmat ovat monesti niin monimutkaisia, ettei niihin ole mahdollista selvittää optimaalista ratkaisua. Geneettisessä ohjelmassa ja geneettisessä algoritmissa läsnä oleva satunnaisuus auttaa välttämään tyypillisiä ongelmia, joihin deterministiset optimointimenetelmät törmäävät, kuten funktion paikalliset minimit. Kuviossa 8 on piirretty optimoitavan funktion kuvaaja punaisella ja sinisellä on merkitty piste, jossa optimointialgoritmi on nyt. Optimoitavan funktion luonne ei ole tiedossa

optimointialgoritmillä eli algoritmi ei voi matemaattisesti laskea parasta tulosta, joka tässä olisi pienin mahdollinen y-arvo. Sinisen pisteen koordinaateista katsoen molempiin suuntiin x-akselin suunnassa tilanne huononee eli funktion arvo kasvaa. Mutaation kautta tapahtuva satunnainen muutos geneettisessä algoritmissa tai ohjelmassa auttaa pääsemään pois paikallisista minimikohdista.



KUVIO 8. Optimoitava funktio ja nykyinen paras tiedossa oleva ratkaisu

Perusvaiheet GP:n ohjelman suorituksessa ovat seuraavat:

1. Populaatio alustetaan sattumanvaraisilla ratkaisuvaihtoehdoilla.
2. Jokainen populaation jäsen arvioidaan.
3. Valitaan parhaat yksilöt paritukseen ja seuraavaan sukupolveen.
4. Poistetaan huonot yksilöt.
5. Palaataan kohtaan 2 kunnes lopettamisehto on saavutettu. Lopettamisehto voi olla tietyn suuruinen tulos tai jokin määrä testattuja sukupolvia.
6. Palautetaan paras yksilö.

Geneettisen algoritmin ja ohjelman merkittävin ero on, että geneettisessä algoritmissa ratkaisumalli on koodattu geeniksi ja täten ratkaisu on yleensä ennalta määrätyn pituinen ja muotoinen. Geneettisessä ohjelmassa ratkaisulle on yleensä määritelty maksimipituus, mutta lopullisen ratkaisun pituus tai muoto ei ole tiedossa. Tämä ero selviää hyvin myöhemmin, kun vakiojako-ongelmaa selvitetään geneettisen algoritmin avulla ja jako-ohjelmaa geneettisen ohjelman avulla.

Evoluutio-oppimisessa ratkaisuvaihtoehtoja kutsutaan kromosomeiksi, jotka ovat erilaisia merkkijonoja päätetystä koodaustavasta riippuen. Kromosomit koostuvat geeneistä, jotka voivat saada jonkun arvon tai tyypin. Alleelit kuvaavat geeneille mahdollisia arvoja tai tyyppejä.

3.3.1 Jälkeläisten valintatavat

Evoluutio-oppimisessa jälkeläisten valinta on erittäin suuressa roolissa, kun tarkastellaan ongelmien ratkaisukykyä. Jälkeläisten valinnassa tulisi onnistua valitsemaan sopivimmat yksilöt seuraaviin sukupolviin, mutta kuitenkin niin, että populaatiossa säilyy riittävän paljon erilaisia ratkaisuvaihtoehtoja.

Rulettimallissa koko populaatio ajatellaan rulettipyöräksi, jota pyöritetään, kun valitaan paritukseen sopivia vanhempia. Jokaisella populaation kromosomilla on oma-alue ruletissa, joka määritellään kyseisen yksilön hyvyyden suhteena populaation keskiarvoon. Näin paremmilla yksilöillä on suurempi mahdollisuus tulla valituksi kuin huonommilla yksilöillä, mutta mikä tahansa yksilöistä voidaan valita. (Affenzeller, Wagner, Winker & Beham 2009, 6.)

Lineaarisessa valinnassa kaikki populaation jäsenet järjestetään niiden hyvyyden suhteen jonoksi, jossa parhaat yksilöt monistetaan (esimääritellyn) n kertaa useammin kuin huonot yksilöt ja näin parhaat todennäköisemmin tulevat valituksi seuraavaan populaatioon.
Turnausvalintamenetelmiä on käytössä useita erilaisia. Useimmiten käytetty on n-turnausmenetelmä, jossa n-yksilöä valitaan populaatiosta ja paras näistä valitaan jälkeläisten tuotantoon.

3.3.2 Paritustavat

Yhden kohdan parituksessa paritettavasta kromosomista valitaan satunnainen kohta, jonka jälkeen olevat osat korvataan toisesta paritettavasta kromosomista saatavilla tiedoilla. Näin voidaan tuottaa yksi tai kaksi uutta kromosomia populaatioon.

Ensimmäinen uusi kromosomi muodostuu 1. isännän alusta ja 2. isännän lopusta. Toinen uusi kromosomi muodostuu 1. isännän lopusta ja 2. isännän alusta.

Monen kohdan parituksessa on vastaavanlainen ajatus kuin yhden kohdan parituksessa, mutta katkaisukohtia on n kappaletta. Joidenkin tutkijoiden mukaan monen kohdan paritus toimii parhaiten erilaisten pitkien merkkijonojen parhaiden puolien yhdistäjänä (Affenzeller ym. 2009, 8.)

3.3.3 Populaation muodostustavat parituksen jälkeen

Täydellisessä korvauksessa koko alkuperäinen populaatio korvataan uusilla, parituksesta saaduilla kromosomeilla. Tämä voi aiheuttaa populaation parhaiden kromosomien tuhoutumista, mutta toisaalta voi tehokkaasti estää populaation jumiutumista paikalliseen minimiin.

Elitismissä alkuperäisestä populaatiosta säilytetään n kappaletta parhaita kromosomeja seuraavaan populaation ja mahdollistetaan näin parhaille yksilöille teoriassa kuolemattomuus. Joskus parhaat yksilöt voidaan kuitenkin valita mutaation kohteeksi parituksen jälkeen, jolloin kuolemattomuutta ei esiinny. Mutaation olessa mahdollinen käytetään nimitystä ”heikko elitimismi”.

Vanhasta populaatiosta voidaan tuhota n kappaletta huonoimpia tai satunnaisesti valittuja yksilöitä, jotka korvataan uusilla parituksesta saaduilla yksilöillä. Turnausvalinnassa testataan uudet yksilöt ja verrataan niitä jo olemassa oleviin yksilöihin. Parhaimmat pääsevät mukaan uuteen populaatioon, josta seuraava paritus suoritetaan.

3.4 Koneoppimismenetelmän valinta

Päätöksentekopuiden tai neuroverkkojen käyttäminen tässä työssä esillä olevan ongelman ratkaisuun vaatisi hyvän ratkaisumallin tuntemista, joka voitaisiin sitten muuttaa konekieliseksi päätöksentekopuun tai neuroverkon avulla. Linjan operaat-

toreita ja muita työntekijöitä haastateltaessa ei löytynyt niin yksityiskohtaista toimintaodotusta, että opettamista olisi voitu suorittaa. Harkinnassa oli myös tapa, jossa ihminen ohjaisi jakolaitteita painikkeilta ja neuroverkko pyrkisi omaksumaan toiminnasta säännöt jokaiselle jakolaitteelle. Linjalla olevien näköesteiden vuoksi ei kuitenkaan ole mahdollista yhdestä kohdasta nähdä ja ohjata kaikkia jakolaitteita ja puristimia.

Haastatteluiden pohjalta päädyttiin selvittämään toimintamallia, jossa viilut jaetaan kiinteällä järjesteyksellä puristimille. Tähän toimintamalliin geneettisen algoritmin kromosomit oli helppo koodata ja geneettistä algoritmia lähdettiin kehittämään ensisijaisena vaihtoehtona. Toimintamallissa havaittiin myöhemmin ei-toivottuja ominaisuuksia ja siksi muita ratkaisuvaihtoehtoja selvitettiin geneettisellä ohjelmalla.

4 .NET-OHJELMOINTIYMPÄRISTÖ

Ohjelmistokehitys tehtiin Visual Studiossa Microsoftin .NET ohjelmointiympäristössä. Visual Studion 2008 versiolla voi kehittää sovelluksia .NET 2.0 – 3.5 -versioille ja näistä käyttöön valittiin 3.5. Ohjelmointikielenä käytettiin geneettisen algoritmin osalta Visual Basicia ja geneettisen ohjelman osalta C#:aa. Microsofin .NET -ohjelmointiympäristö tukee noin 20:tä ohjelmointikieltä, joista eniten käytettyjä ovat C# ja Visual Basic. Kirjastoista löytyvillä toiminnoilla voidaan suorittaa suurin osa ohjelmistojen vaatimista toiminnoista, jolloin ohjelmoija voi keskittyä enemmän ohjelmiston businesslogiikkaan. Kirjastojen lisäksi .NET:n toinen merkittävä osa on CLR (Common Language Runtime) eli ajoympäristö. CLR kääntää esikäännetyn ohjelman käyttöjärjestelmän ymmärrettävään muotoon suorituksen aikana tarjoten mahdollisuuden prosessorityyppikohtaiseen optimointiin ilman erillistä ohjelman kääntöä jokaiselle prosessorityypille. Ohjelman suorituksen aikaisen käännön vuoksi ensimmäisen kerran kutakin ohjelmalohkoa suoritettaessa on ohjelman suoritus hieman normaalia hitaampi. Aikakriittisissä sovelluksissa tämän voi joutua huomioimaan, mutta normaaleissa sovelluksissa käyttäjä ei huomaa eroa. (Wikipedia 2011.)

5 MITSUBISHI-KOMMUNIKAATIO

Sovelluksen tulee osata keskustella linjan logiikan kanssa, joka tässä tapauksessa on Mitsubishin Q-sarjan logiikka. Logiikassa on mahdollisuus USB- tai ethernet-kommunikointiin. Logiikka oli jo valmiiksi yhdistetty tehtaan ethernet-verkkoon, joten se valikoitui käytettäväksi yhteydeksi. Saatuina etuina verrattuna USB:hen on esimerkiksi yhteyden nopeus ja se, että sovellusta suorittavan PC:n ei tarvitse sijaita linjan läheisyydessä. Kommunikointi voitaisiin käytännössä hoitaa joko kaupallisella sovelluksella eli IO-serverillä tai tekemällä kommunikointi itse.

5.1 Kaupallinen IO-serveri

Kaupalliset IO-serverit maksavat joitakin satoja euroja, mutta yleensä kuitenkin vähemmän kuin 1000 euroa. Kaupallisen IO-serverin hyviä puolia ovat, että ne osaavat yleensä enemmän kuin yhden fyysisen rajapinnan ja logiikkamallin. Ohjelma on myös hyvin testattu ja harvoin löytyy sellaisia virheitä, jotka estävät ohjelman toiminnan.

Huonoihin puoliin kuuluu ehdottomasti se, kun jokin virhe löytyy, on sen etsiminen, tarkkarajaaminen ja kiertäminen tai korjaaminen käytännössä hyvin aikaa vievää. Aiemmissa kokemuksissa on myös havaittu, että jotkin IO-serverit katkaisevat logiikkayhteyden aina keskustelyryhmää vaihtaessaan eivätkä näin yllä parhaaseen mahdolliseen suorituskykyyn.

5.2 Itse tehty kommunikointiluokka

Mitsubishin Q-sarja käyttää ethernetissä MC3E-protokollaa, jonka materiaalin Mitsubishi jakaa ilmaiseksi Internetissä. Tämä mahdollistaa oman kommunikointiluokan tekemisen juuri niillä ominaisuuksilla kuin halutaan. Luokan ohjelmointi vaatii PC-ohjelmointitaitoa ja kyseisen logiikkamallin tuntemusta. Yleensä vaatimuksiin kuuluu myös jonkinlainen testausmahdollisuus. Ilman testiympäristöä ei voida varmistua ohjelman toimivuudesta myös poikkeustilanteissa, kuten verkkokatkokset.

Kommunikointiluokan ohjelmoinnissa on työtä huomattavasti enemmän kuin kaupallisen sovelluksen käyttämisessä, mutta kommunikointiluokka jää talteen myöhempää käyttöä varten. Tässä työssä samaa kommunikointiluokkaa käytettiin sekä tiedonkeruuohjelmissa, että varsinaisessa GP-ohjelmassa.

6 SIMULAATIOMALLIT

Simulaatio on erittäin käytetty, ellei käytetyin, keino toiminnan tutkimisessa. Järjestelmän ajatellaan olevan kokoelma esimerkiksi ihmisiä tai koneita, jotka toimivat vuorovaikutuksessa toistensa kanssa saavuttaakseen tavoitteen. (Law 2007, 2.)

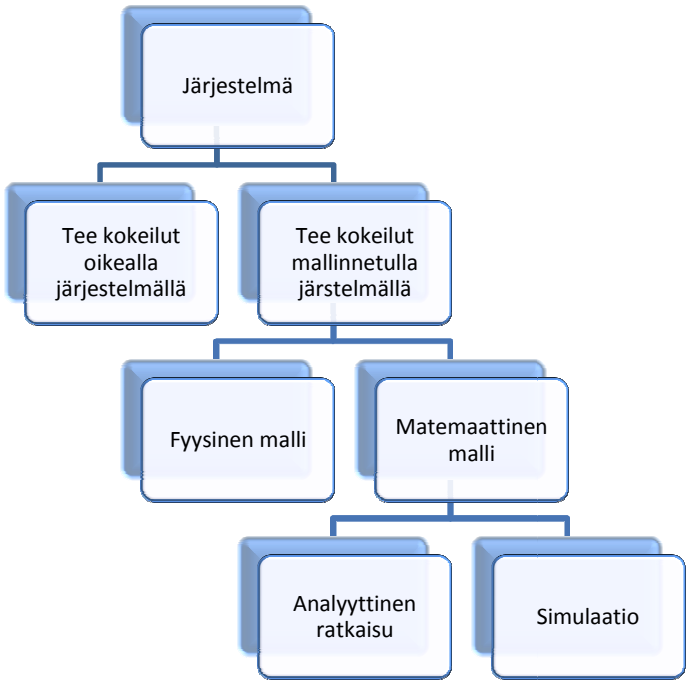
Järjestelmänä toimii tässä työssä viilujen jatkamislinjan loppuosa, jossa viilut jaetaan puristimille. Tuotantokapasiteetti on erittäinen arvokasta, ja siksi päädyttiin luomaan linjasta simulaatiomallin, jossa toimintaa voi visuaalisesti havainnoida.

Silloin kun on mahdollista tehdä kokeet oikealla järjestelmällä, kannattaa niin yleensä tehdä. Näin voidaan varmistaa, että tutkimustulokset ovat oikeita. Harvoin kuitenkaan on mahdollista suorittaa kokeita oikealla järjestelmällä, koska se tulisi hyvin kalliiksi tai häiritsi järjestelmää paljon. Mallia käytettäessä tulee aina varmistua mallin oikeellisuudesta. (Law 2007, 3.)

Toisinaan on järkevää luoda pienoismalli, tai oikeankokoinen malli, jostakin järjestelmästä jolloin sitä voidaan mielekkäämmin tutkia. Käytännössä suurin osa tehdyistä malleista on matemaattisia, jotka esittävät järjestelmää loogisten ja määrällisten yhteyksien muodossa. Arvoja muuttamalla mallista nähdään, miten järjestelmä reagoi, mikäli malli on todenmukainen. (Law 2007, 3.)

Mikäli matemaattinen malli on yksinkertainen, voi olla mahdollista ratkaista halutut asiat ja niiden väliset suhteet matemaattisista yhtälöistä. Usein kuitenkin joko järjestelmän toiminta tai asioiden väliset suhteet ovat hyvin monimutkaisia ja näin ollen analyyttinen ratkaisu ei ole mahdollista. Tällöin mallia tutkitaan simulaation avulla eli annetaan mallille tuloihin erilaisia arvoja ja katsotaan mitä mallin ulostulossa tapahtuu. Kuviossa 9 on esitetty järjestelmän erilaiset tutkimisvaihtoehdot. (Law 2007, 3.)

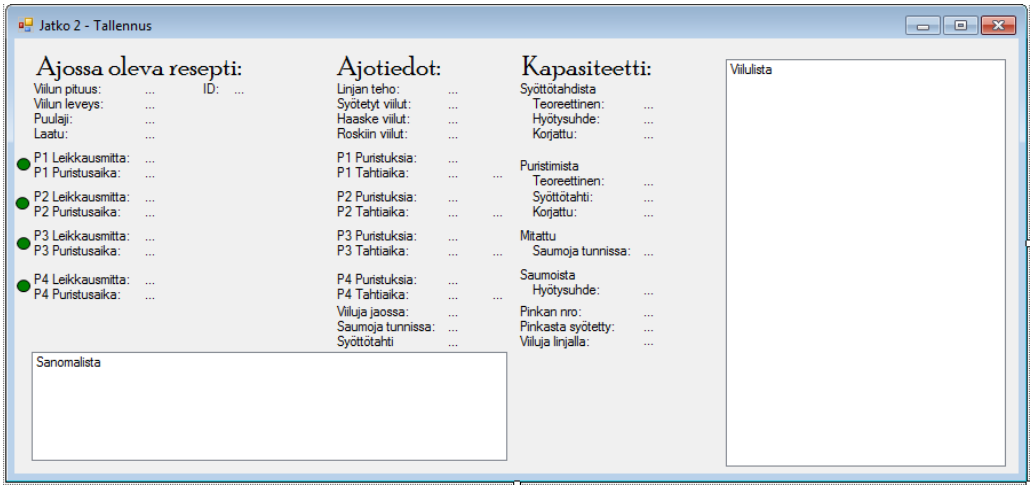
Staattisessa simulointimallissa malli ei muutu ajan suhteen tai aika ei yksinkertaisesti vaikuta malliin. Dynaamisessa mallissa aika vaikuttaa simulaatiomallin tilaan, kuten esimerkiksi kuljetinjärjestelmässä. (Law 2007, 3.)



KUVIO 9. Tapoja tutkia järjestelmää (Law 2007.)

7 ESISELVITYKSET

Linjalle tehtiin erilaisia tiedonkeruita kun linjan toimintaa ja muuttuvia avainparametreja etsittiin työn alkuvaiheessa. Tallennettavat tiedot luettiin logiikasta käyttäen itse tehtyä MC3E-luokkaa ja tallennettiin sopivin väliajoin palvelimelle Microsoftin SQL-Server 2005 -tietokantaan. Kuviossa 10 on tiedonkeruuohjelman pääsivu, jossa voi seurata tallennuksen aikana linjan ajoparametreja ja toimintaa.



KUVIO 10. Tiedonkeruuohjelman pääsivu

7.1 Reseptikohtainen tiedonkeruu

Reseptikohtaisen tiedonkeruun pohjana oli ajatus siitä, että ajettaessa linjalla eri asetuksilla, voitaisiin toimia eri tavalla. Nämä tavat voisi operaattori määritellä ja näillä tiedoilla voitaisiin opettaa neuraaliverkkoja haluttu ja tarvittu määrä, ja käytössä olevaa verkkoa vaihdettaisiin tarpeen mukaan. Ongelmaksi voisi muodostua vaihtokohdat, mutta toisaalta, jos neuraaliverkko onnistuisi yleistämään linjan toimintaa riittävästi, ei ongelmia muodostuisi.

Column Name	Data Type	Allow Nulls
ID	bigint	☐
Aikaleima	datetime	☑
ViilunPituus	int	☑
ViilunLeveys	int	☑
Puulaji	nvarchar(50)	☑
Laatu	nvarchar(50)	☑
P1Leikkausmitta	decimal(3, 1)	☑
P1Puristusaika	decimal(3, 1)	☑
P2Leikkausmitta	decimal(3, 1)	☑
P2Puristusaika	decimal(3, 1)	☑
P3Leikkausmitta	decimal(3, 1)	☑
P3Puristusaika	decimal(3, 1)	☑
P4Leikkausmitta	decimal(3, 1)	☑
P4Puristusaika	decimal(3, 1)	☑
P1Ajossa	bit	☑
P2Ajossa	bit	☑
P3Ajossa	bit	☑
P4Ajossa	bit	☑

KUVIO 11. Reseptitiedonkeruun taulun rakenne

Reseptitauluun valittiin kaikki ne tiedot, jotka oleellisesti vaikuttavat viilujen syöttöön ja jakoon linjalla. Tallennettavat tiedot näkyvät kuviossa 11, kuten myös tietokantataulun rakenne. Tallennusohjelma tallentaa uuden reseptin, mikäli ajossa oleva kombinaatiota ei ole vielä tietokannassa kun samanlainen resepti löytyy tietokannasta, käytetään tietojen tallennukseen sen reseptin ID:tä. Jokainen ajettu resepti löytyy siis reseptitaulusta vain kerran ja omaa uniikin ID-numeron. Reseptitiedonkeruuta pidettiin päällä noin kuukausi, ja sinä aikana tietokantaan tuli yli 500 kappaletta erilaisia ajoasetuksia (LIITE 2). Tällainen määrä erilaisia

ajoasetuksia on mahdottoman suuri ja niinpä resepteihin pohjautuva neuraaliverkkojen opetus lopetettiin liian työläänä ratkaisuna.

7.2 Syöttöpinokohtainen tiedonkeruu

Syöttöpinokohtaisessa tiedonkeruussa seurattiin eri syöttöpinojen välisiä eroja. Mikäli syöttötahtia korjattaisiin sillä määrällä viiluja, joka poistuu linjalta, tulisi luultavasti ongelmaksi suuret vaihtelut syötettävän viilun laadussa.

Column Name	Data Type	Allow Nulls
ID	bigint	☐
Aikaleima	datetime	☑
ViilunPituus	int	☑
ViilunLeveys	int	☑
Puulaji	nvarchar(50)	☑
Laatu	nvarchar(50)	☑
SyotetytViilut	int	☑
HaaskeViilut	int	☑
RoskiinViilut	int	☑
P1Puristuksia	int	☑
P2Puristuksia	int	☑
P3Puristuksia	int	☑
P4Puristuksia	int	☑

KUVIO 12. Syöttöpinotaulun rakenne

Haettaessa tallennetuista tiedoista linjalta haaskeeseen menneiden viilujen osuus voidaan nähdä hyvin suuret vaihteluvälit eri kuormien välillä. Ottaen huomioon, että syöttökuorman vaihtojen välillä on 20–30 minuuttia, täytyisi linjalla tapahtuvan laskennan reagoida nopeasti syöttökuorman vaihdon jälkeen ja sen jälkeen hitaammin, jotta säätö voisi toimia. Kuviossa 12 näkyy tietokantataulun rakenne, jonka perusteella muodostetulla SQL-kyselyllä (kuvio 13) saadaan tuloksena haaskeprosentit syöttöpinoittain (kuvio 14).

```
SELECT    Aikaleima
        , ViilunLeveys
        , CONVERT(decimal(10, 1),
            CONVERT(decimal(10, 1),
              HaaskeViilut + RoskiinViilut) * 100 / SyotetytViilut
        ) AS HaaskePros
FROM      Pinkkakeruu
WHERE     (SyotetytViilut > 0) AND (ViilunLeveys = 1310)
ORDER BY Aikaleima
```

KUVIO 13. Linjalta poistuneiden viilujen hakeminen tietokannasta

Aikaleima	ViilunLeveys	HaaskePros
22.7.2010 14:29:30	1310	9,7
22.7.2010 14:52:06	1310	4,3
22.7.2010 15:11:25	1310	4,0
22.7.2010 15:52:28	1310	11,8
22.7.2010 16:10:53	1310	8,4
22.7.2010 16:32:28	1310	4,7
22.7.2010 17:15:53	1310	7,2
22.7.2010 18:06:40	1310	5,0
22.7.2010 18:37:01	1310	7,6
22.7.2010 19:10:23	1310	7,6
22.7.2010 19:59:02	1310	7,3
22.7.2010 20:00:46	1310	5,1
22.7.2010 20:37:53	1310	5,1
22.7.2010 20:52:04	1310	6,6
22.7.2010 20:53:19	1310	10,2
23.7.2010 6:01:38	1310	53,8
23.7.2010 6:29:28	1310	9,1
23.7.2010 6:34:01	1310	6,5
23.7.2010 7:00:41	1310	13,7

KUVIO 14. Syöttöpinoittain tietokannasta haettu haaskeprosentti

7.3 Viilujen seuranta

Viilujen seurantatiedonkeruu tehtiin selvittämään viilulaskurin toimintaa. Viilulaskurilla oli tarkoitus seurata jaossa olevien viilujen määrää. Viilulaskurilla olisi voitu ohjata viilujen syöttötahtia nopeammaksi, kun viilujen määrä jaossa vähenee. Viilulaskurin toiminta-ajatus oli, että laskuriin lasketaan +1 aina, kun viilu menee jakokuljettimelle, ja -1 aina, kun joku puristimista ottaa viilun jakokuljettimilta. Tämän lisäksi viilulaskuri asetettiin nollaksi aina kun kuljettimet pyörivät yhden minuutin tyhjänä. Käytännössä tällainen tilanne tulee jokaisessa syöttö-

kuorman vaihdossa eli noin 20 - 30 minuutin välein. Tiedontallennustahti oli viisi sekuntia, koska yhden viilun ajaminen jakoon kestää noin kaksi sekuntia. Näin tallennustahdin aikana ei ehdi tapahtua suuria muutoksia viilulaskurissa. Viilulaskurin arvo tulee virheelliseksi siinä kohtaa, kun käyttäjä poistaa käsin viilun joltakin kuljettimelta. Tällaista virhettä on vaikea ohjelmallisesti korjata ja saada ohjelma toimimaan varmasti erilaisissa tilanteissa. Tiedonkeruulla tämän virheen suuruutta arvioitiin ja päätettiin olla kehittämättä tämän laskurin ja sitä kautta syöttötahdin ohjausta.

	Column Name	Data Type	Allow Nulls
🔑	ID	bigint	☐
	Aikaleima	datetime	☑
	Resepti	int	☑
	LinjanTeho	int	☑
	ViilujaLinjalla	int	☑
▶			☐

KUVIO 15. Viilujen seurannan tietokantataulun rakenne

ID	Aikaleima	Resepti	LinjanTeho	ViilujaLinjalla
5	28.7.2010 7:54:19	420	75	0
6	28.7.2010 7:54:24	420	75	2
7	28.7.2010 7:54:29	420	75	5
8	28.7.2010 7:54:35	420	75	5
9	28.7.2010 7:54:40	420	75	6
10	28.7.2010 7:54:46	420	75	6
11	28.7.2010 7:54:51	420	75	5
12	28.7.2010 7:54:57	422	75	3
13	28.7.2010 7:55:02	422	100	4
14	28.7.2010 7:55:08	420	100	0
15	28.7.2010 7:55:13	420	100	-2
16	28.7.2010 7:55:19	420	100	-3
17	28.7.2010 7:55:24	420	75	-3
18	28.7.2010 7:55:30	420	75	-2
19	28.7.2010 7:55:35	420	75	-1
20	28.7.2010 7:55:41	420	0	2
21	28.7.2010 7:55:46	420	50	4
22	28.7.2010 7:55:52	420	50	5
23	28.7.2010 7:55:57	420	50	4
24	28.7.2010 7:56:03	420	50	4
25	28.7.2010 7:56:08	420	50	3
26	28.7.2010 7:56:14	420	75	2

KUVIO 16. Viilujen seurannan tallentamia tietoja

Kuviossa 15 näkyy tietokantataulun rakenne. Tietoja kerättiin kolmelta vuorokaudelta eli yhteensä yhdeksältä työvuorolta, ja näistä tiedoista käy ilmi (kuvio 16), että hyvin usein jaossa on näennäisesti liikaa viiluja eli linjan operaattorit ovat poistaneet viilun käsin tai viilussa olevien reikien suodatus on epäonnistunut ja ohjelma on laskenut saman viilun kahteen kertaan. Yhdenkin viilun virhe viilulaskurissa aiheuttaisi huomattavan korjauksen syöttötahdissa ja tätä virhettä olisi vaikea huomata ennen kuin linja olisi täynnä viiluja ja kuljettimet pysähtyisivät.

8 GENEETTISEN ALGORITMIN LUONTI

8.1 Toimintaperiaate

Ensimmäinen linjasimulaation luonti tehtiin Siemensin S7 -logiikalla ja siihen liitettiin PC-sovellus, joka raportoi erän tulokset ja antoi logiikalle uuden kromosomin ajoon. Tässä kokeilussa testattiin vakiojaon teoriaa, joka tuli esille keskusteluissa operaattorien ja esimiesten kanssa. Siemens S7 -logiikkaan rakennettiin ohjelmallisesti linjan simulaatiomalli ja siirtorekistereissä siirreltiin viiluja virtuaalilinjalla. PC-valvomo tehtiin Visual Basic -ohjelmointikielellä .NET -ympäristössä ja logiikkakommunikointiin käytettiin avoimen lähdekoodin projektia nimeltä libnodave. PC-sovelluksessa oli valvomon lisäksi geneettinen algoritmi, joka muodosti halutun mittaisia kromosomeja, joissa geenien alleelit vastasivat puristimen numeroa. Esimerkiksi siis 4-1-3-2 kromosomin viilut jaettiin siten, että ensimmäinen viilu ohjattiin puristimelle 4, seuraava puristimelle 1 ja niin edelleen. Viiluille annettiin numerot ennen ensimmäistä jakolaitetta ja sen jälkeen viilun numeroa siirrettiin eteenpäin varastopaikoille sitä mukaa kun viilu eteni linjalla kohti puristinta ja matkalla jakolaitteita ohjattiin sen mukaan.

8.2 Logiikkaohjelma

Logiikkaohjelmaa luodessa rakenne kopioitiin suoraan linjalla ajossa olevasta logiikasta, koska linjan simulaatiomallin piti mahdollisimman hyvin vastata oikeaa toimintaa. Puristinten mallinnuksessa pyrittiin mallintamaan viilun leikkausmitan vaikutus puristimen tahtiaikaan, vaikkakaan tällä ei lopputuloksen kannalta ollut merkitystä.

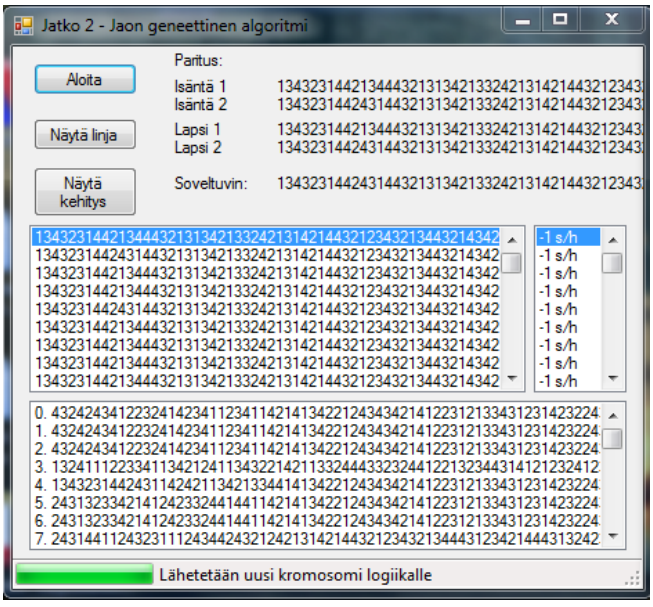
Ohjelma on jaettu toiminnoittain lohkoihin, joista tärkeimmät ovat

- FC5 jakolaitteiden tilatiedot
- FC2 moottorien ohjaus
- FC6 puristimet
- FB50 jakolaitteiden ohjaus
- FC1 viilupuskurit
- OB1 pääohjelma ja ajastimien arvojen skaalaukset.

PC-ohjelma lähettää logiikalle tiettyjä parametreja, joiden mukaan logiikka osaa ajaa erän itsenäisesti ja ilmoittaa tulokset PC:lle ajon valmistuttua. Logiikalle lähetetään mm. ajettavien viilujen kappalemäärä, jakojärjestestys ja haluttaessa satunnaistettu viilujen syöttöväli.

8.3 PC-ohjelma

Tietokoneohjelma, joka hoitaa logiikan käskytystä, geneettistä algoritmia ja tulosten raportointia, on rakenteeltaan yksinkertainen. Sovelluksen runko suoritetaan gaetusivu-tiedostossa, joka hoitaa logiikkakommunikoinnin libnodave-kirjaston kautta ja geneettisen algoritmin. Kuviossa 17 näkyy ohjelman pääsivu. Ohjelmaa suoritetaan ajastimen ohjaamana, ja mikäli sukupolvessa kaikilla kromosomeilla ei ole vielä arvoa, lähetetään niitä yksikerrallaan logiikalle arvioitavaksi. Hyvyysfunktio on erässä saavutettu kapasiteetti, jossa suurempi on parempi ja maksimikapasiteetti määräytyy syöttötahdin mukaan. Jako on riittävän hyvä kun koko syöttökapasiteetti saadaan ajetuksi ilman keskeytyksiä.

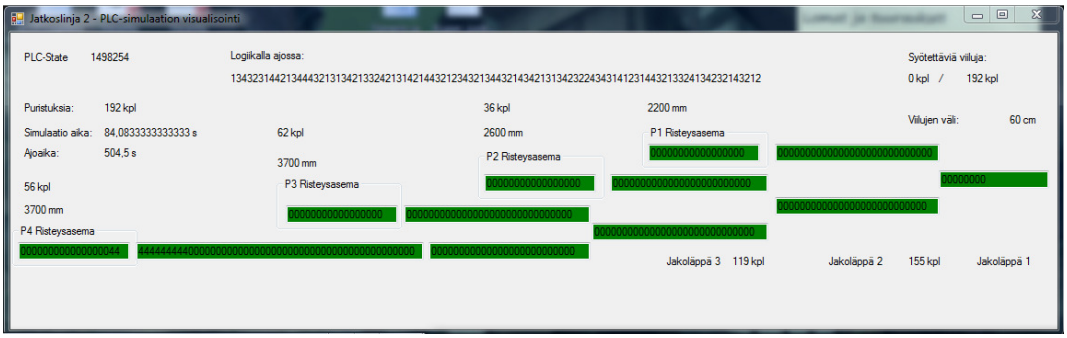


KUVIO 17. Geneettisen algoritmin pääsivu

Populaatiosta puolet korvataan parituksesta saaduilla kromosomeilla. Uuteen populaatioon säilyvät kromosomit valitaan heikko elitismi -menetelmää käyttäen lähinnä sen yksinkertaisuuden vuoksi. Populaatiosta poistetaan puolet kaikista

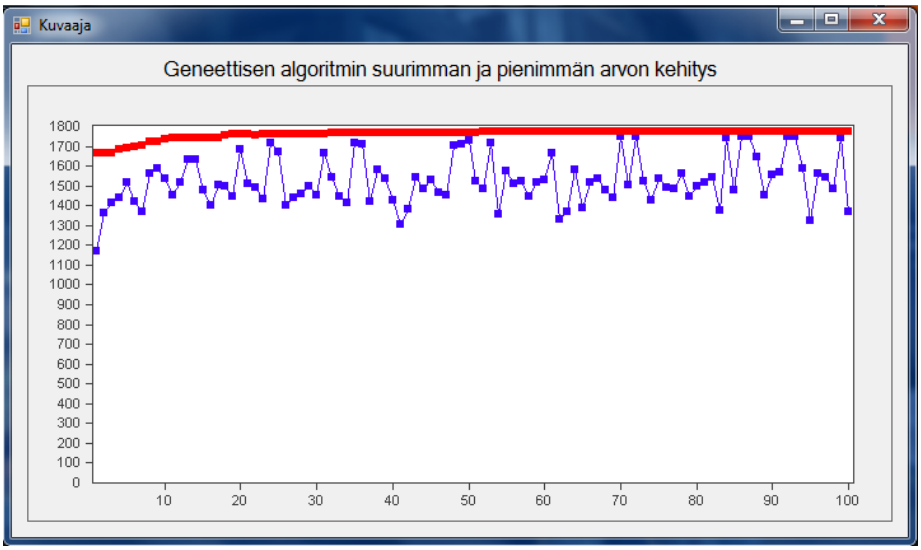
kromosomeista ja nämä valitaan puhtaasti niiden huonouden perusteella. Paritu-ksen jälkeen on mahdollista, että jokin kromosomeista valitaan mutatoitavaksi, jolloin jokainen geeni on mahdollista vaihtaa toiseksi (tai samaksi). Mahdollisen mutaation jälkeen kaikki uudet kromosomit arvioidaan ja sama sykli toistuu uudestaan.

Linjasimulaation oikeellisuus tarkistettiin visuaalisesti käyttäen tilatietonäkymää, joka aukeaa pääsivulta näytä linja -painikkeesta. Kuviossa 18 on tilatietonäkymä, johon luetaan kuljettimien ja varastopaikkojen tilat logiikasta.



KUVIO 18. Geneettisen algoritmin tilatietonäkymä. Kuviossa viilut liikkuvat oikealta vasemmalle

Ohjelma raportoi algoritmin kehityksen pääsivun alempaan listaan, jossa näkyy kunkin sukupolven paras kromosomi. Tämän lisäksi kehitystä voi seurata kuvaajasivusta, joka aukeaa pääsivulta näytä linja -painiketta painamalla. Kuviossa 19 näkyy tyypillinen kehitys. Kuvasta nähdään, että hyvin pienellä sukupolvimäärällä on saavutettu suurin mahdollinen kapasiteetti (punainen viiva) ja että paritus sekä mutaatio ovat luoneet uusia, erilaisia ratkaisuvaihtoehtoja, koska sukupolven heikon arvo (violetti viiva) on vaihdellut sukupolvien välillä.

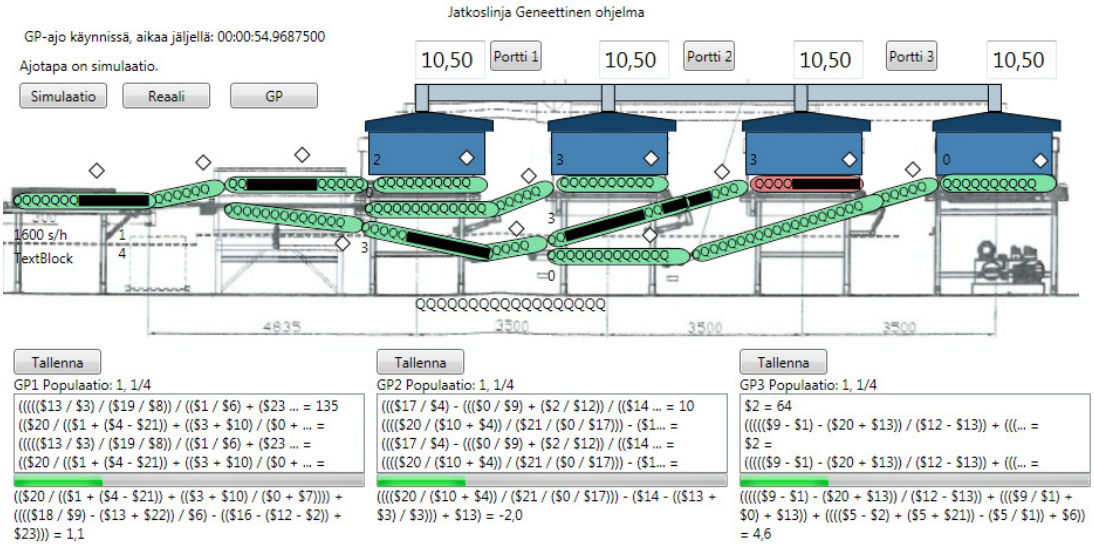


KUVIO 19. Geneettisen algoritmin kehitys sukupolvittain. Punainen on sukupolven parhaan kromosomin arvo ja violetti sukupolven huonoimman kromosomin arvo.

9 GENEETTISEN OHJELMAN LUONTI

9.1 Toiminta ja käyttöliittymä

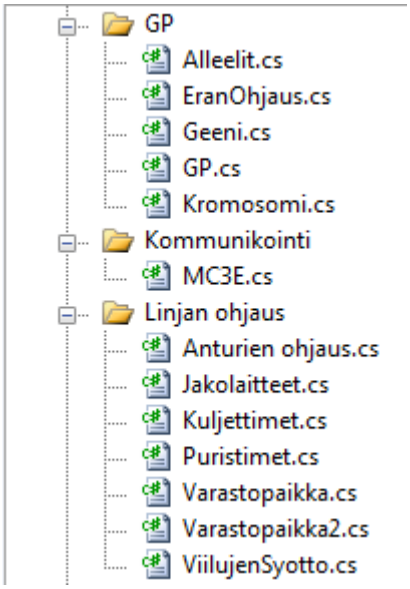
Geneettinen ohjelma luotiin Microsoftin VC#.Net -ohjelmointikielellä Visual Studio -kehitysympäristössä. Ohjelmasta haluttiin tehdä aiempaa visuaalisempi toiminnan tarkastamisen helpottamiseksi. Kuviossa 20 nähdään kuva ohjelman pääsivusta ja siitä voidaan erottaa muutamat oleelliset kohdat. Vasemmalla ylhäällä on eri ajotapojen valintapainikkeet, keskellä ja oikealla ylhäällä ovat puristimen tahtiajat sekä porttien kytkimet. Portti on nouseva kuljetin juuri ennen puristimia kaksi, kolme ja neljä, joista linjan operaattori voi kulkea linjan ollessa käynnissä. Portit otettiin mukaan simulaation siksi, että erilaisia koeajoja tehdessä voidaan yksi puristin ottaa pois ajosta portilla ja katsoa, miten geneettinen ohjelma jatkaa kehitystä. Sivun keskivaiheilla on linjan simulaatio-osuus, jossa viilujen ja jakolaitteiden liikkeet on helppo havainnoida. Pienet vaaleat salmiakit kuvaavat antureita ja näyttävät reaaliajotilassa valokennojen todelliset tilat. Sivun alalaidassa on jokaiselle jakolaitteelle oma geneettinen ohjelma ja näkyvissä on nykyisen sukupolven kromosomit. Sukupolvi on mahdollista tallentaa seuraavan koeajon alkupopulaatioksi tallenna-painikkeesta.



KUVIO 20. Geneettisen ohjelman pääsivu

9.2 Ohjelman rakenne

Geneettinen ohjelma on rakennettu kokonaisuudessaan Jatkoslinja2GP -nimitilaan. Toiminnalliset osat on jaettu useaan tiedostoon ja luokkaan. Kuviossa 21 näkyy ohjelman rakenne tiedostoittain. Tiedostoissa olevat luokat seuraavat tiedostojen nimiä.



KUVIO 21. Geneettisen ohjelman tiedostojen kansiorakenne.

Tärkeimpiä luokkia ja tiedostoja ovat seuraavat:

- Window1 – Ensimmäisenä ladattava luokka jossa tehdään muiden luokkien alustukset ja käyttöliittymän päivitys.
- MC3E – Mitsubishi-kommunikointi varten luotu luokka, joka itsenäisesti päivittää logiikasta haetut tiedot Datalista nimiseen muuttujaan.
- GP – Geneettinen ohjelma, josta avataan jokaiselle jakolaitteelle oma instanssi.
- Erän ohjaus on toteutettu osana Window1-luokkaa ja sen pääasiallisena tehtävän on hoitaa tiedonsiirto käyttöliittymän ja geneettisen ohjelman välillä.
- Linjan ohjaus on toteutettu osana Window1-luokkaa ja se toimii linjan simulaatiomallina sekä näyttää todelliset tiedot reaaliajotilassa.

9.3 Ajotavat

Ohjelmaa suunniteltaessa ajatuksena oli tehdä kolme ajotapaa ohjelmaan. Näitä olisivat simulaatiotila, reaalitila ja GP. Simulaatiotilassa malliin syötetään kuvitteellisia viiluja halutulla syöttötahdilla säännöllisin väliajoin. Tällä tavalla geneet-

tinen ohjelma etsii ratkaisua normaaliin ajotilanteeseen, kun viiluja tulee tasaisesti ja ne kaikki pitää saada jaettua puristimille tasaisesti parhaan tuotannon saavuttamiseksi. Reaalitilassa linjalla näkyy todellinen tilanne eli se, missä viiluja milloinkin on (simulaatiomallin puskureissa), antureiden sekä toimilaitteiden tilat. Reaalitilaa käytettiin simulaatiomallin testaukseen. Reaalitilassa näkee jaon kokonaistilanteen paljon paremmin kuin paikan päällä, koska linjalla on paljon näköesteistä eikä koko tilannetta voi havainnoida yhdestä paikasta. GP-ajotavassa oli tarkoituksena, että simulaatiomalliin tulevat viilut olisivat linjalla olevia todellisia viiluja, jotka siis tulisivat epäsäännöllisesti riippuen syöttötahdista, haaske- ja roskamääristä sekä kaikista ongelmista tuotannon aikana. GP-ajotavassa haasteena on hyvyysfunktion osalta hetket, jolloin tuotantoa ei ole yritettykään tehdä. GP-ajovaihtoehtoa ei kuitenkaan viimeistelty ja testattu, koska geneettisellä ohjelmalla ei löydetty hyviinkään olosuhteisiin riittävän hyvää ratkaisua.

9.4 Logiikkayhteys

Tuotantolinjalla on käytössä Mitsubishin Q-sarjan logiikka, joka osaa keskustella ethernet-verkon kautta mm. Mitsubishin 3E -protokollalla. Mitsubishin protokollamäärittelyiden pohjalta luotiin MC3E-luokka, joka osaa keskustella yhden logiikan kanssa, lukea ja kirjoittaa muistipaikkoja sekä datarekisterejä. Luokkaa voidaan myöhemmin käyttää muissa .NET -ohjelmointiympäristöllä tehtävissä projekteissa. Luokkaan ei tehty tiedonsiirtotaulua, jollainen löytyy kaupallisista versioista. Näin kommunikoinnin optimointi on puhtaasti ohjelmoijan harteilla, koska anturien tietoja halutaan päivittää reaaliajotilassa huomattavasti nopeammin kuin tuotannon laskureita tai muita vastaavia tietoja.

9.5 Tuotantolinjasimulaatio

Simulaatiomalli koostuu pääasiassa kuljettimia mallintavista siirtorekistereistä. Näiden lisäksi on kolme jakolaitetta, kolme kulkuporttia ennen puristimia sekä neljä puristinta. Kaikki neljä puristinta eivät ole todellisuudessa täysin samanlaisia, koska vanhoissa puristimissa (1-3) on ajan tuomaa kulumaa ja puristin 4 on

vasta muutaman vuoden vanha. Simulaatiomallin yksinkertaistamiseksi kaikki puristimet mallinnettiin samanlaisiksi. Tällä yksinkertaistuksella ei ole kuitenkaan jaon kannalta juurikaan merkitystä, vaan nämä tahtiaikaerot vaikuttavat enemmän syöttötahtiin.

Windows Presentation Foundation eli WPF-ohjelmoinnissa on mahdollista sitoa animaatiot osaksi ohjelmaa ja tällä saavutetaan huomattava etu simulaatiomallin yhtenäisyydessä animaatioiden ja ohjelmankierron välillä. Ohjelma todellakin odottaa jakolaitteiden säädetyn animaatioajan verran, eikä ole riskiä siitä, että animaatiolla ja ohjelmassa olevalla odotuksella olisi eri kestoaika. Näin animaation tiloista luotavat tapahtumat toimivat samaan tapaan kuin liikkeissä mahdollisesti olevat rajakytkimet. Simulaatiomallia luotaessa käytettiin hyväksi tiedonkeruusta saatuja tietoja, joista nähtiin puristimien keskimääräinen tahtiaika. Tahtiaika on yksinkertaistetussa puristimien mallissa vakio. Jakolaitteiden liikeaikoja ei, vaan ne, kuten myös joitakin muita aikoja selvitettiin sekuntikellolla linjalta. Simulaatiomallin toiminnan testaus tehtiin pääasiassa reaaliajotilassa, jossa kaikki ohjelmointivirheet tulivatkin hyvin esiin. Näin saatiin myös kuljettimien käyntiehdot, jakolaitteiden liike-ehdot ja muut asiat vastaamaan mahdollisimman hyvin linjan logiikkaohjelmaa.

9.6 Jakolaitteiden geneettiset ohjelmat

Toteutustapoja on kaksi erilaista; tehdä yksi geneettinen ohjelma ja etsiä kaikille jakolaitteille sopivaa, yleispätevää, ratkaisua tai tehdä jokaiselle oman geneettinen ohjelma ja mahdollistaa erilaisiin tuloksiin päätyminen. Tekemällä jokaiselle jakolaiteelle oman geneettisen ohjelman voitiin jokaiselle jakolaitteelle antaa erilainen määrä parametreja. Jakolaiteelle yksi annettiin useampia parametreja, koska oletettiin sen toimintalogiikan olevan monimutkaisempi kuin muiden.

Ohjelmaan tehtiin myös mahdollisuus ottaa joitakin geneettisiä ohjelmia pois käytöstä. Näin voidaan esimerkiksi yhdellä jakolaitteella ajaa hyväksi havaittua ratkaisua ja etsiä kahdella muulla toimivaa ratkaisua.

9.6.1 Hyvyys-funktio

Kromosomien arviointi funktiossa kokeiltaan muutamia erilaisia vaihtoehtoja, joista lopullisesti päädyttiin versioon, jossa jakolaitteelle lasketaan virhepisteitä. Virhepisteitä saa, kun jakolaitteen kuljetin pysähtyy, jos ylhäällä tai alhaalla oleva kuljetin pyörii. Hyvyys-funktio olettaa, että tällöin viilut on jaettu väärin ja aiheutettu tilanne, jossa yhdelle puristimelle tai puristin ryhmälle on mennyt liian monta viilua. Virhepisteitä saa myös epätasaisesta jaosta. Jakolaitteen 1 pitäisi siis jakaa puristimelle 1 ja muille tasan suhteessa ¼, ettei virhepisteitä tulisi. Hyvyys-funktio ei siis ota kantaa saavutettuun kapasiteettiin, mikä oli yksi alkuperäisistä tavoitteista ja geneettisen algoritmin hyvyys-funktiossa käytetty tapa. Kapasiteettiajatuksesta luovuttiin, koska simulaatiossa haluttiin ajaa useilla eri kapasiteeteillä oikean toiminnan varmistamiseksi. Geneettisen ohjelman oppimiskykyä testattaessa havaittiin, että alkupopulaatiosta saa parhaiten erotettua huonot ajamalla hieman liian suurella kapasiteetilla, jolloin kaikki virheet korostuvat. Mikään jako-ohjelma ei voi syöttää puristimia nopeammin kuin mitä ne voivat viiluja käsitellä, ja tällöin pysähdysvirhepisteitä tulee paljon. Viilujen tasaisen jaon virhepisteet eivät ole merkittävässä osassa ensimmäisissä sukupolvissa. Viimeiset sukupolvet, jotka ajetaan normaalilla kapasiteetilla, erotellaan tasaisuuden mukaan.

Kolme peräkkäistä geneettistä ohjelmaa aiheuttivat myös ongelmia pisteiden jakamisessa, kun ensimmäinen jakolaite ei jakanut yhtään viilua alas, vaan antoi kaikki viilut puristimille 1. Tällöin jakolaitteen 1 kromosomi sai huonot pisteet, mutta normaalilla laskutavalla jakolaitteiden 2 ja 3 kromosomit saivat 0 virhepistettä, koska ne eivät jakaneet epätasaisesti eivätkä pysäyttäneet kuljettimia kertaakaan. Tästä syystä ohjelmaan piti tehdä ehto, että jakolaitteen 2 tulos arvostellaan vain, kun sen kautta on kulkenut vähintään 50 % viiluista ja jakolaitteelle 3 rajaarvo on 30 %. Tämä aiheuttaa jakolaitteille erilaisen sukupolvien kehittymistahdin, mutta estää vääränlaisten johtopäätösten teon.

9.6.2 Erän laskenta

Kaikki population kromosomit ajetaan simulaatiossa ja tätä ajoa kutsutaan eräksi. Yhden erän kesto valitaan halutun pituiseksi, yleisesti erän kesto oli 2 – 5 minuuttia. Erään liittyy oleellisesti virhepisteiden laskenta hyvyys-funktiolle. Virhepisteistä laskettiin jakolaitteelle, mikäli jakolaite on pysäyttänyt sille tuovan kuljettimen ja ainakin toinen jakolaitteelta lähtevistä kuljettimista pyörii. Tällöin jakolaite saa yhden virhepisteen aina sekunnin pysähdystä kohti. Erittäin lasketaan myös puristimien puristukset eli tehdyt saumat. Puristusten määrästä lasketaan jakosuhde, jota hyvyys-funktio käyttää jakolaitteen sakottamiseen.

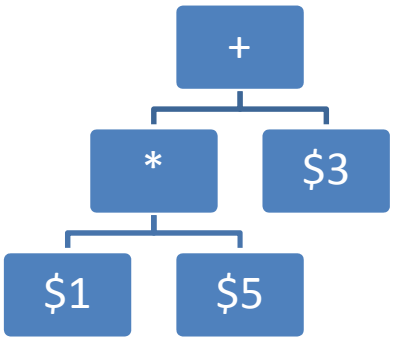
9.6.3 Jakolaitteiden ohjaus

Jakolaitteita ohjataan kromosomin arvon mukaan siten, että tulos, joka on alle raja-arvon (säädettävä parametri), ohjaa jakolaitteen alas. Suurempi tulos kuin raja-arvo ohjaa jakolaitteen ylös. Raja-arvoparametrin arvo koeajoissa oli 0, eli negatiivisilla tuloksissa jakolaite ohjataan alas ja positiivisilla ylös. Jakolaitteen liikettä rajoitetaan ohjelmasta vain silloin, kun viilu on jakolaitteen ja seuraavan kuljettimen välissä. Käytännössä tällainen tilanne tulee estää, koska se aiheuttaisi viilun tuhoutumisen.

9.6.4 Geenit

Geeni voi olla kahta eri tyyppiä, joko tekijä tai laskutoimitus. Tekijä on jokin geneettiselle ohjelmalle annetuista parametreista, kuten tieto siitä, näkeekö jakolaitteen anturi viilun vai ei. Laskutoimitus on jokin ennalta määritellyistä matemaattisista laskutoimituksista. Geenillä voi olla lapsia nollasta kahteen. Mikäli geenin tyyppi on tekijä, sillä ei ole lapsia ollenkaan ja geenin arvo viittaa siihen parametriin, joka geenin kohdalle sijoitetaan kromosomin arvoa laskettaessa. Laskutoimitus tyyppisellä geenillä on joko yksi tai kaksi lasta, riippuen laskutoimituksesta. Esimerkiksi yhteenlaskulla on aina kaksi lasta ja neliöjuurella vain yksi. Geenin lapset ovat geenejä, joilla on kaikki samat mahdollisuudet. Nimitys lapsi tulee siitä, että ne sijaitsevat kaavapuussa alemmalla tasolla kuin isäntägeeni. Kuviossa

22 on esitetty kromosomi (($1 * $5) + $3) kaavapuuna. (Poli, Langdon & McPhee 2008, 10.)



KUVIO 22. Kromosomien kaavapuuesitys

Geenityypit:

```
enum GeeniTyyppi
{
    Lasku,
    Parametri
}
```

Mahdolliset laskutoimitukset:

```
enum LaskuTyyppi
{
    Yhteenlasku,
    Vahennyslasku,
    Jakolasku,
    Kertolasku,
    Neliojuuri,
    Potenssi,
    Sini,
    Kosini,
    Tangentti
}
```

Laskutoimituksista rajattiin alustavien koeajojen jälkeen pois trigonometriset funktiot, mutta nämä laskutoimitukset ovat olemassa geneettisen ohjelman luokassa.

9.7 Kaavojen muodostus

9.7.1 Kaavapuut

Geneettinen ohjelma muodostaa populaation aluksi sattumanvaraisia kromosomeja eli matemaattisia lausekkeita, joilla ohjataan jakolaitteiden toimintaa siten, että positiivinen tulos on merkki ylösohjauksesta ja negatiivinen tulos alasohjauksesta. Tuloksella nolla jakolaitetta ei liikuteta. Kaavan muodostus tapahtuu seuraavalla tavalla jokaiselle populaation kromosomille. Kromosomin ensimmäiseksi geeniksi valitaan satunnaisesti jokin mahdollisista laskutoimituksista, pelkkä tekijä ei ole mahdollinen. Jokainen laskutoimitus on yhtä mahdollinen ensimmäisenä geeninä. Tämän jälkeen tarkastellaan, kuinka monta tekijää tässä laskutoimituksessa on ja luodaan sille tarvittavat tekijägeenit. Tekijägeeneillä on 1/3 mahdollisuus olla parametri ja 2/3 mahdollisuus olla laskutoimitus, mutta kuitenkin niin, että alimmalla mahdollisella tasolla annetaan aina tekijäksi parametri. Näin varmistutaan siitä, ettei algoritmi tuota äärettömän pitkiä laskutoimituksia. Kaikilla parametreilla on yhtä suuri mahdollisuus tulla valituksi tekijäksi sen jälkeen, kun tekijän tyypiksi on valittu parametri.

9.7.2 Epäsäännöllisten puiden merkitys

Tämänlaisessa kaavassa, jossa tekijöillä voi olla hyvin erilainen merkitys lopputulokseen, on tärkeää, etteivät kaavapuut ole symmetrisiä. Tällöin jotkin haarat, jotka jäävät lähemmäksi alkua, omaavat suuremman merkityksen kuin ne, jotka ovat kaukana alusta. Eräissä muissa ongelmissa, kuten esimerkiksi binaarikoodauksen selvityksessä kaikki bitit ovat yhtä tärkeitä oikean lopputuloksen saamiseksi.

9.7.3 Kaavojen lukeminen

Geneettinen ohjelma muodostaa jakolaitekohtaisesti kromosomeja eli matemaattisia lausekkeita. Esimerkiksi: ($3 + ((($17 - $20) + ($9 - $21)) / ($1 + ($2 / $23))) - $15)). Laskutoimitukset suoritetaan kuten normaalistikin, mutta kuitenkin niin, että ei-sallitut laskutoimitukset (kuten nollalla jako) on estetty. Nollalla jako -tilanteessa luku jaetaan nollan sijasta numerolla yksi. Dollarimerkillä alkavat luvut ovat geneettisen ohjelman käytössä olevia parametreja, luku on nolla indeksinä esitetty.

Liitteestä 3 löytyvät parametritaulukot, joista näkyy jakolaitteille annetut parametrit. Annettujen parametrien lisäksi geneettisen ohjelman luokkaan koodattiin kiinteiksi parametreiksi vakioluvut 0, 1, -1, 0.5, 0,5, 0.1 ja -0.1.

10 SYÖTTÖTAHDIN AUTOMATISOINTI

Syöttötahtia kokeiltiin automatisoida sen välittömän kapasiteettivaikutuksen vuoksi sen jälkeen kun jaon kapasiteettiongelma oli saatu ratkaistua. Linjalla on paljon muuttuvia asioita, jotka vaikuttavat puristimien tahtiaikaan. Merkittäviä tahtiaikaan vaikuttavia tekijöitä ovat ajo- ja laatuparametrit, kuten ajossa olevan viilun leveys ja puristusaika. Nämä tekijät operaattorit osaavat ennakoida ja voivat säätää syöttötahtia sopivaksi ajoparametreihin. Puristimissa huollon tai kulumien mukaan muuttuvia tekijöitä, kuten toimilaitteiden nopeus, hihnojen kitka ja muut vastavaat, operaattorit eivät osaa ennakoida. Operaattorit eivät myöskään osaa ennakoida näiden muutoksien vaikutusta syöttötahtitarpeeseen. Syöttötahti säädetään logiikassa 0,01 sekunnin tarkkuudella, mutta operaattorit valitsevat sopivimman kytkimen kolmesta asennosta. Nämä kytkimen asennot valitsevat erikokoisille viiluille esivalituista parametreista syöttötahdin. Tämäntyyppinen säätö ei anna mahdollisuutta minkäänlaiseen hienosäätöön, ja siksi käytännössä ajetaan aina liian pienellä syöttötahdilla.

Syöttötahdin automatisointia varten linjan logiikkaan tehtiin ohjelma, joka laskee puristimien tahtiajoista keskiarvon, muuntaa sen viiluiksi tunnissa ja laskee siitä sopivan syöttötahdin. Tähän olisi mahdollista myös lisätä korjausta haaskeen ja roskien kompensoinniksi, mutta se voi olla vaikea saada tasapainoon.

Linjalla syöttötahdin automaattisäätöä kokeiltaessa havaittiin, että vakiojaolla ajettaessa puristimien tahtiajoista laskettava keskiarvo ei toiminut erityisen hyvin. Syöttötahdin nopeuden laskentakaava sinällään on oikein ja vaikuttaa toimivalta, mutta kaavan mukaan laskettu syöttötahti oli koko ajan hieman liian suuri. Tämä johtuu siitä, että puristimien nopeuksissa on eroja ja keskiarvollinen tahtiaika sekä vakiojako yhdessä aiheuttavat ongelman. Viiluja pitäisi syöttää hitaimman puristimen mukaan, ettei kyseinen puristin ruuhkautuisi. Hitaimmain puristimen mukaan syötettäessä on mahdollista, että kun nopeuserot ovat suuret, ei linjalle saada tarpeeksi viiluja ja täysi kapasiteetti ei toteudu kaikilla puristimilla. Automaattinen syöttötahdin ohjelmaa muutettiin sisältämään kaksi laskentatapaa, yhteensä ja hitaimman mukaan. Liittessä 4 on kuva operointipäätteeseen tehdystä sivusta, josta näkyy näiden laskentatapojen ero käytännössä.

11 TULOKSET

11.1 Geneettinen algoritmi

11.1.1 Koejärjestely

Vakiojakoa pidettiin alkuperäisessä suunnitelmassa parhaana vaihtoehtona, koska näin voitaisiin aina taata tasainen viilun jakautuminen puristimille. Parasta vakiojakojärjestystä etsittiin geneettisellä algoritmilla ja Siemens S7 -logiikassa suoritetussa linjan simulaatiomallissa. Geneettisessä algoritmissa ajettiin eripituisia kromosomeja neljästä merkistä aina 40 merkkiin. Kaikki pituudet olivat jaollisia neljällä. Kromosomien geeneillä oli neljä alleellia, jotka olivat kokonaisluvut 1 … 4. Luku esittää sitä puristinta, jolle viilun halutaan menevän. Jokaiselle jakoon tulevalle viilulle siis annettiin luku 1 … 4 ja linjasimulaatio piti huolen siitä, että viilu ajettiin puristimelle vaikka, se pysäyttäisi kuljettimet. Puristinten tahtiajat säädettiin vastaamaan 1700 saumaa tunnissa kapasiteettia, mikä on hyvin lähellä todellista maksimikapasiteettia. Kaikilla puristimilla oli kuitenkin yksilöllinen tahtiaika, joka on seurausta ajettavan tuotteen ominaisuuksista. Tämä ero tahtiajassa ei ole suuri, mutta asettaa rajoituksia linjan maksimikapasiteetille. Viiluja syötettiin jokaiselle kromosomille 192 kappaletta ja mitattiin näiden ajamiseen kulunut aika. Ajasta ja syötettyjen viilujen määrästä laskettiin kapasiteetti, jota käytettiin hyvyys-funktiona ja jota siis algoritmi pyrki maksimoimaan.

11.1.2 Koeajojen tulokset

Koeajomääräksi valittiin riittävän suuri määrä viiluja, ettei puristimelle 4 oleva pidempi matka vaikuttanut tuloksiin. Ajettaessa vain yksi viilu olisi selvää, että algoritmi valitsisi puristimen 1 mielummin kuin puristimen 4. Tämä johtuu siitä, että ajanotto alkaa, kun viilu syötetään linjalle, ja loppuu, kun viilu poistuu puristimelta. Koeajoissa ajettiin useita toistoja ja kaikissa havaittiin, että 1700 saumaa tunnissa kapasiteetti saavutettiin jo 10 sukupolven aikana populaation alustuksesta. Tulokset eivät olleet mitenkään yhtenäisiä, mistä voitiin päätellä, että viilujen

jakojärjestys ei vaikuta linjalla saavutettavaan kapasiteettiin kunhan jakokuviossa jokainen puristin esiintyy yhtä monta kertaa. Liitteessä 5 on yhteenveto saaduista tuloksista.

11.1.3 Vakiojaon teko linjalle

Geneettisen algoritmin koeajon perusteella linjan jako-ohjelma muutettiin vakiojaoksi ja logiikkaohjelmaan tehtiin tarvittavat siirtorekisterit viilujen seuraamiselle. Ohjelmaa tehdessä sovittiin, että kaikki viilut, joilla ei ole numeroa (ohjelma ei pystynyt seuraamaan niitä) tai jotka eivät pääse haluamalleen puristimelle (puristin ei ole valmiina kun viilu on kohdalla), ajetaan puristimelle numero 4. Puristimella 4 on eniten varastotilaa, ja sen kapasiteetti on mekaanisesn rakenteen takia eniten riippuvainen viilujen tulemisesta puristimelle.

Viilujen seuraaminen kuljettimilla ei ole täysin ongelmatonta ottaen huomioon viilujen liukumisen hihnoilla ja niiden kolme eri päämittaa. Jakolaitteiden ja porttien liikkeet vielä hieman sekoittavat ohjelmaa, koska viilut saattavat liukua pois valokennoilta liikkeen aikana. Ohjelmaa ja varastopaikkoja koeajettaessa päästiin hyvin toimintavarmaan ohjelmaan luomalla varastopaikkoja noin puolikkaan viilun välein, vaikka valokennoja ei olekaan niin tiheästi. Viilun etenemisen seurantaan käytettiin erilaisia ajastimia, koska ratapulssiantureita ei ole saatavilla. Linjalle tehty logiikkaohjelma löytyy liitteestä 7.

11.1.4 Tulokset

Tammikuussa 2009 vakiojako-ohjelma tehtiin linjan logiikkaan ja toimintaa seurattiin kevään ajan. Vakiojaon tekemisen jälkeen linjan operaattoreilta kerätty palaute oli hyvin positiivista ja lähes kaikki jako-ongelmat olivat poistuneet. Viilujen jakautuminen puristimille oli myös hyvin tasaista, ja vaikutti siltä, että valtaosa ongelmista oli ratkaistu.

Pidempään tuotantoajoa seuratessa selvisi, että viilujen vakiojaossa on omat ongelmansa. Viilujen lähes täydellinen tasaisesti jakaminen aiheuttaa sen, että syöttötahdin on oltava säädetty hyvin lähelle hitaimman nopeutta, ettei puristimelle numero 4 ala kertyä liikaa viiluja. Kapasiteetin kannalta tämä jako ei siis ole täysin ideaalinen, kun jotakin puristinta on jouduttu hidastamaan mekaanisten tai sähköisten ongelmien takia. Puristimet 1, 2 ja 3 ovat vanhoja, ja niiden mekaniikka on kohtuullisen kulunutta, mikä aiheuttaa ennustamattomia eroja tahtiaikaan. Lisäksi puristin 4 on varustettu erilaisella mekaanisella rakenteella ja tahtiaika on sitä kautta erilainen kuin muissa puristimissa. Oletettavaa siis on, että vakiojaolla ei saada suurinta mahdollista kapasiteettia tällä linjalla.

11.2 Geneettinen ohjelma

11.2.1 Koejärjestelyt

Koeajoja suoritettiin tietokoneohjelmalla, johon oli tehty sekä geneettinen ohjelma että linjasimulaatio. Geneettinen ohjelma luotiin siten, että kromosomit ovat epäsäännöllisiä ja eripituisia matemaattisia lausekkeita. Populaation koko vaihteli koeajoissa 4 ja 20:n välillä ja populaatiosta paritettiin yhden kohdan parituksella 50 %. Seuraavaan sukupolveen valittiin kromosomit käyttämällä heikkoa elitismiä. Sukupolvia ajettiin enemmän kuin sata yhdessä koeajossa. Linjasimulaatioon tehtiin kaksi vaihtoehtoa jakolaitteiden ohjaukselle. Ensimmäisessä vaihtoehdossa kromosomin arvo lasketaan vain kerran ja se päätös pitää, kunnes uusi viilu tulee jakolaitteelle. Toisessa vaihtoehdossa kromosomin arvoa lasketaan koko ajan ja jakolaite saa liikkua aina, kun kuljettimien välissä ei ole viilua. Viilujen syöttökapasiteetti valittiin 400 ja 1700:n väliltä ja ohjelma osaa vaihdella kahden syöttökapasiteetin välillä saman koeajon aikana. Hyvyys-funktiona oli aiemmin esitelty virhepisteiden laskenta, jonka arvoa siis geneettinen ohjelma pyrki minimoimaan.

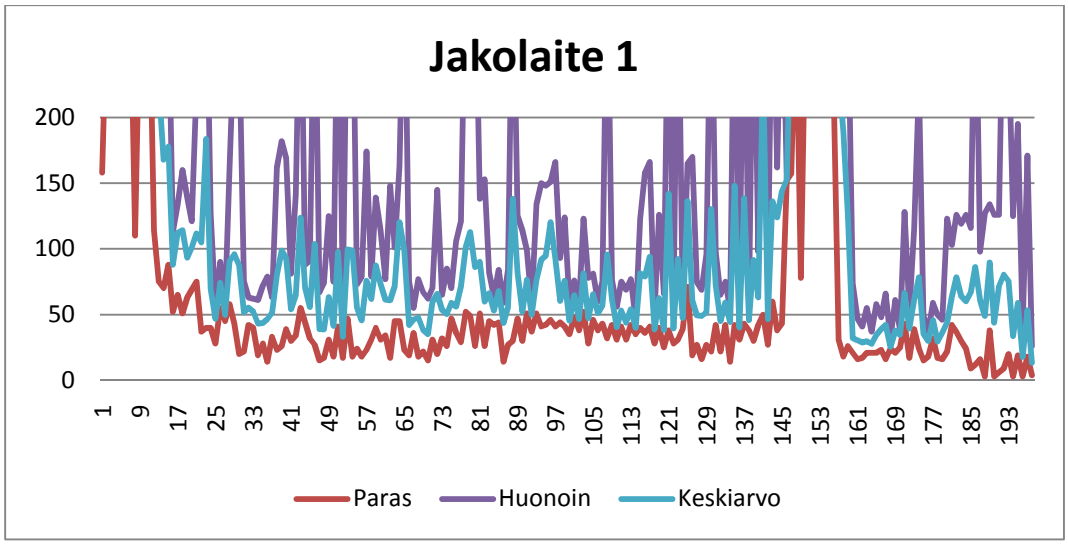
Myöhemmässä vaiheessa yksi koeajo suoritettiin myös niin, että aiemmista koeajoista parhaat valittiin käsin tähän alkupopulaation jokaiselle jakolaitteelle erik-

seen. Alkupopulaation koko oli tällöin 12 kappaletta ja sama kromosomi ei esiintynyt kahdesti missään alkupopulaatiossa.

11.2.2 Koeajojen tulokset

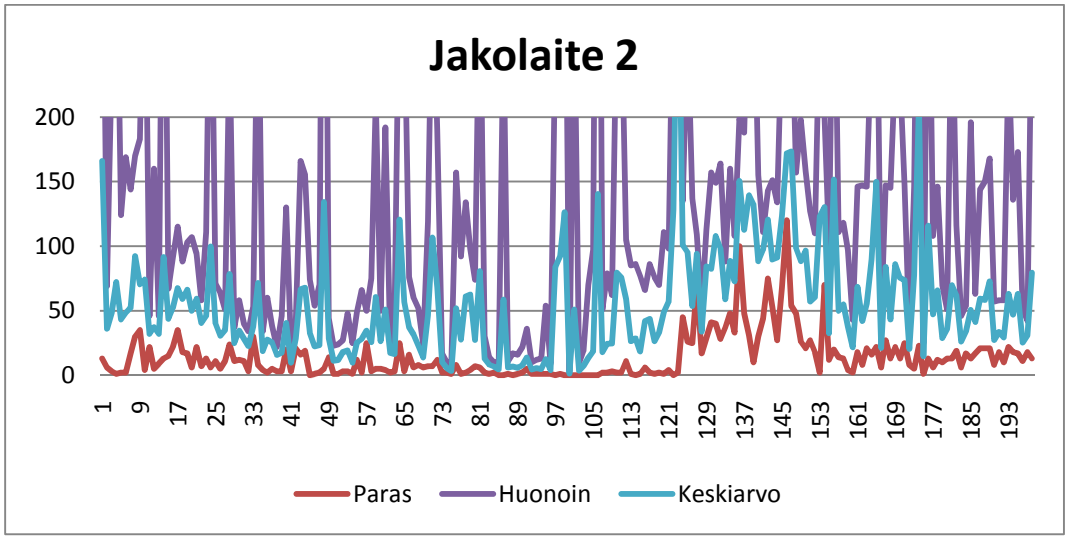
Seuraavassa esitetään yhden koeajon tulokset, mutta muiden koeajojen tulokset ovat olleet yhteneväiset näiden tulosten kanssa vaikka geneettisten ohjelmien parametreja on muuteltu. Kaikkien kolmen jakolaitteen tulokset on koostettu kuviohin. Kuvioissa punaisella värillä on merkitty kunkin sukupolven parhaan kromosomin arvo, violetilla huonoimman kromosomin arvo ja sinisellä populaation kromosomien keskiarvo. Kuvaajissa pienempi arvo on parempi. Tarkemmat tulokset löytyvät liitteestä 8. Huomattavaa on, että sukupolvet eivät etene samassa tahdissa, koska alussa jakolaite yksi ei välttämättä jaa viiluja ollenkaan jakolaitteille kaksi ja kolme.

Jakolaitteen yksi tulokset ovat näkyvissä kuviossa 23. Kuvaajasta nähdään, että ensimmäisten kahdenkymmenen sukupolven aikana tapahtuu merkittävää oppimista, jonka jälkeen tilanne tasaantuu. Seuraavan sadan sukupolven aikana ei tapahdu merkittävää kehitystä tuloksissa, kunnes tulokset heikkenevät merkittävästi. Joidenkin sukupolvien ajan tulokset ovat erittäin huonoja, kunnes tilanne kohenee ja lopulta löytyy ratkaisuja joissa, virhepisteitä ei tule. Populaation huonoimman kromosomin arvosta voidaan havaita ohjelman hakevan erilaisia ratkaisuja vaihtoehtoja joka sukupolvella.



KUVIO 23. Jakolaitteen 1 tulokset

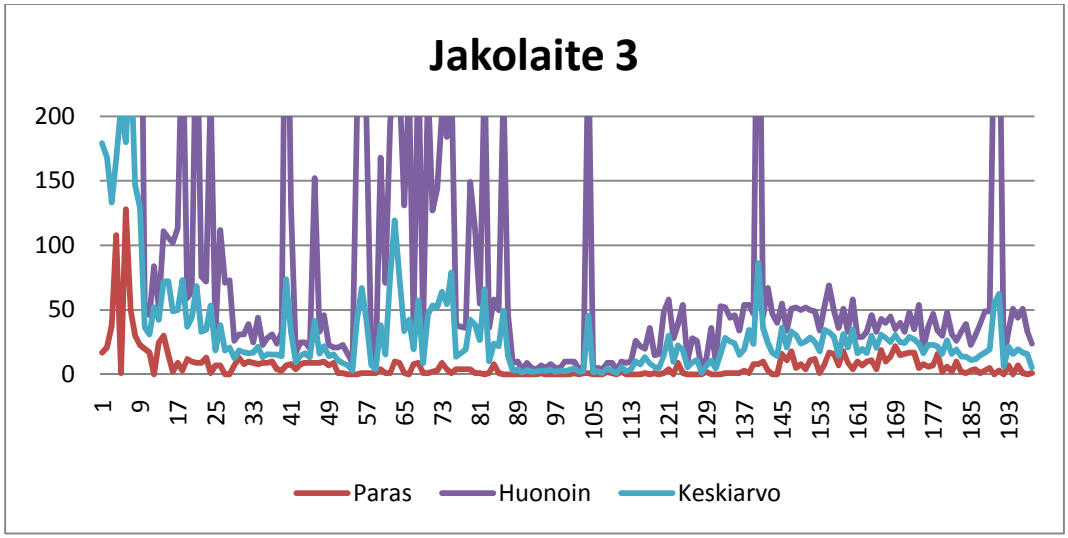
Jakolaitteen kaksi kuvaajasta (kuvio 24) nähdään, että heti alkuperäisessä populaatiossa on mukana hyviä ja huonoja ratkaisuvaihtoehtoja. Ensimmäisten muutaman kymmenen sukupolven aikana osa selvästi huonoista vaihtoehdoista karsiutuu pois, jonka jälkeen merkittäviä muutoksia ei tapahdu. Sadannen sukupolven kohdalla tulokset ovat erittäin hyviä, joka voi johtua siitä, että jakolaite 1 ei toiminut niillä sukupolvilla kunnolla tai sitten kyseisten sukupolvien kromosomit ovat hyviä.



KUVIO 24. Jakolaitteen 2 tulokset

Jakolaitteen kolme kuvaajasta (kuvio 25) nähdään, että alkupopulaatiosta asti mukana on hyviä ratkaisuja. Hirveästi huonoja ratkaisuja ei ole populaatioissa, joka

selittää sukupolven huonoimman kromosomin suhteellisen hyvät arvot. Jakolaitteen kolme jako on yksinkertaisempi kuin muut, koska jakolaite jakaa vain kahdelle puristimelle.



Kuvio 25. Jakolaitteen 3 tulokset

Kaikki jakolaitteet pääsevät yksinään tyydyttäviin tuloksiin, mutta yhdessä jokaisen jakolaitteen heikkoudet korostuvat. Simulaatiossa kokeiltiin ajaa käsin valittuja (aiempien koeajojen parhaita) kromosomeja yhdessä, mutta tulokset eivät merkittävästi parantuneet. Ennen linjalle ajoon siirtoa tavoitteena olisi kuitenkin pystyä ajamaan kaikki nopeudet 400 ja 1600:n välillä ilman keskeytyksiä ja niin, että jako olisi tasainen kun kaikki puristimet ovat ajossa. Kromosomeja, jotka täyttävät nämä vaatimukset, ei löytynyt, ja siksi koeajojen parhaita ei koettu riittävän hyviksi, että niitä olisi alettu ohjelmoida linjan logiikkaan.

11.3 Tehollinen käyntiaika, kapasiteetti

Opinnäytetyön tavoitteena oli saada nostettua linjan kapasiteettia poistamalla pullonkaula. Liittessä 6 on käyty läpi tuotannollisia mittareita, joista nähdään, että linjan maksimikapasiteetti on nyt riittävällä tasolla opinnäytetyön osana tehdyn vakiojaon seurauksena. Hyvästä maksimikapasiteetista huolimatta vuorossa ei ole keskimäärin saatu aikaiseksi yhtään enempää tuotantoa kuin ennen, mikä ilmenee pitkän ajan seurannasta liitteessä 6.

Linjalla ei investoinnin aikaan ollut käytössä tehollisen käyntiajan mittausta, joten alkutilanteeseen ei voida vertailla. Kuitenkin hyvänä vertailukohtana toimivat tehtaan muut linjat, joissa on vastaavanlainen tehollisen käyntiajan mittari tehtynä. Liittessä 7 on esitetty tämä vertailu yhden normaalin tuotantoviikon osalta. Vertailusta nähdään, että jokainen linja on tehnyt normaalit tulokset eli urakka palkkauksen pohjana oleva suoritekerroin on ollut lähellä yhtä. Silti linjojen tuottavassa ajoajassa on suuria eroja. Tästä nähdään, että saman tuotantomäärän kuin ennen pystyy nykyään tekemään lyhyemmässä ajassa, mutta palkkausjärjestelmä ei kannusta tekemään sitä tuotantomäärää mihin linjalla pystyisi.

12 POHDINTA

Tuloksista voidaan päätellä linjan maksimikapasiteetin olevan nyt oikealla tasolla pullonkaulan poistumisen myötä, mutta tuloksissa ei näy investoinnin tai opinnäytetyössä tehtyjen muutoksien aiheuttaneen minkäänlaista kasvua tuotantomäärissä pitkällä aikavälillä. Tuloksista nähdään haaskeen määrään vähentyneen vuodesta 2008, mikä voi osittain johtua siitä, että linja pysähtelee vähemmän, mutta todennäköisemmin kyse on kuitenkin raaka-aineen parantumisesta. Kapasiteetin kasvamattomuus johtuu osittain taloudellisesta taantumasta ja töiden vähyydestä, mikä koetteli vuosien 2009 ja 2010 aikana. Osittain tuotantomäärien kasvamattomuus saattaa johtua siitä, että nykyinen palkkausjärjestelmä ei riittävästi kannusta tekemään enempää tuotantoa kuin juuri suoritetason verran.

Automaattinen syöttötahti vaikuttaa toimivalta, kun laskentatavaksi on valittu hitaimman puristimen mukaan. Saavutettu kapasiteetti ei ole suurin mahdollinen, mutta kuitenkin suurempi kuin manuaalinen syöttötahdin säätö. Automaattinen syöttötahti ottaa hyvin huomioon eri puulajien ja laatujen erot ja helpottaa siltä osin linjan operaattorien toimintaa. Automaattinen syöttötahdin säätö on käytössä linjalla ja tulevaisuudessa sama muutos tullaan todennäköisesti tekemään tehtaan muillekin jatkoslinjoille. Tulevaisuudessa, kun jatkoslinjoihin saadaan lisää muuttuva nopeuksisia kuljettimia, voidaan tutkia mahdollisuutta kompensoida haaskeen ja roskien aiheuttamaa kapasiteetin menetystä. Automaattinen syöttötahti hitaimman puristimen mukaan ja vakiojakomenetelmä aiheuttavat tilanteen, jossa linjan suurin kapasiteetti saadaan vain, kun kaikki puristimet ovat yhtänopeita. Tuotannonsuunnittelun osalta tämä tarkoittaa sitä, ettei samanaikaisesti ole kapasiteetin kannalta järkevää ajaa reilusti erikokoisia tuotteita. Linjan operaattoreilta erikokoisten tuotteiden ajo mahdollisimman suurella kapasiteetilla vaatii puristimien toiminnan tuntemista niin, että nopeimmat tuotteet osataan sijoittaa ajoon hitaimmille puristimelle ja siten kaventaa puristimien kapasiteettieroja.

Nykyisillä syötön ja viilujen jaon ohjaustavoilla jäädään selvästi teoreettisesta maksimikapasiteetista. Tehtyjen mittausten perusteella kirittävää olisi vielä noin 5 %, mutta se vaatisi ohjaustavan, joka sallii jokaiselle puristimelle yksilöllisen jaon. Geneettinen ohjelma ja linjasimulaatio ovat valmiina .NET -

ohjelmointiympäristössä odottamassa uusia ajatuksia. Tehtyjen tutkimuksien pohjalta voidaan luotettavasti todeta, että viilujen paras mahdollinen jakaminen vaatii keskitetyn päätöksen teon eikä jokainen jakolaite voi toimia itsenäisesti. Keskitetyn päätöksen teon tulisi ottaa huomioon puristimien ja jakolaitteiden lisäksi ainakin haaske ja roskat sekä mahdollisesti myös viilujen syöttö.

Tuotantolinjan toiminnan optimointi on mahdollista toteuttaa koneoppimisen avulla. Yksinkertaisiin ongelmiin kannattaa ratkaisuja etsiä perinteisin keinoin, koska koneoppimisen soveltaminen on verrattaen työlästä ja aikaa vievää. Optimointiongelman tulee olla riittävän monimutkainen, että koneoppimisen soveltaminen tulee kannattavaksi. Koneoppimista voisi käyttää myös toimintamalleja suunniteltaessa, ennen kuin varsinaista linjaa on vielä olemassa. Simulaatiomalli täytyy joka tapauksessa luoda koeajoja varten ja tietokonesimulaatiota on helpompi muokata kuin oikeaa tuotantolinjaa. Tulevaisuudessa monimutkaisten säätöpiirien taustalle tarvitaan malleja, joilla voidaan pidentää säätimen säätöhorisonttia ja näin ennakoida tehtävien säätöjen vaikutusta prosessiin. Koneoppimista voidaan käyttää taustalla olevan mallin päivitykseen todellisesta prosessista mitattujen vasteiden avulla ja näin saavuttaa säästöjä prosessin ohjauksessa välttämällä turhia säätötoimenpiteitä ja parantamalla vasteaikoja.

Koneoppimiselle on paljon sovellusalueita tulevaisuuden teollisuudessa, joissa myös säästöpotentiaali on suuri. Olemassa olevien tuotantolinjojen toimintaa parannettaessa rajoituksia asettaa ensisijassa olemassa oleva mekaniikka. Toisaalta kun ohjausta kehitetään älykkäämmäksi, voidaan joitakin toimenpiteitä ennakoida ja mekaniikkaan kohdistuvat tahtiaikavaatimukset pienenevät. Koneoppimisen avulla voidaan tuotantolinja saada toimimaan oikealla nopeudella annettujen tuote- ja laatuparametrien mukaan, jolloin linjan operaattoreiden työ helpottuu. Uusia linjoja ja sovelluksia kehitettäessä koneoppimisen avulla voidaan saada valmiimpia sovelluksia ensimmäisellä kerralla, vaikka kehitysaika voi pidentyä simulaatiomallien luomisen ja koeajojen suorittamien takia.

LÄHTEET

- Affenzeller, M., Wagner, S., Winker, S. & Beham, A. 2009. Genetic Algorithms and Genetic Programming. CRC Press.

- Koponen, H. 2002. Puulevytuotanto. 3. painos. Helsinki: Edita OY.

- Koskisen OY. Yritys. 2010. [viitattu 27.3.2011].
  Saatavissa: http://www.koskisen.fi/yritys

- Marsland, S. 2009. Machine Learning An Algorithmic Perspective. CRC Press.

- Law, M. 2007. Simulation modeling & analysis. 4.painos. McGraw-Hill.

- Poli, R., Langdon, W. & McPhee, N. 2008. Field Guide to GP

- Wikipedia .NET -ohjelmointiympäristö [viitattu 19.2.2011].
  Saatavissa: http://fi.wikipedia.org/wiki/.NET_Framework